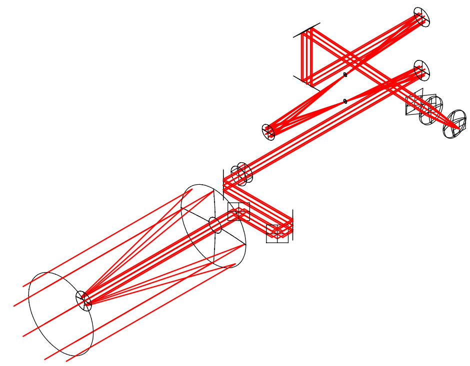


The *Mid*-Infrared
*I*nterferometric
Instrument for the
Very Large Telescope
Interferometer

# MIDI

—

## Observation Software

**User and Maintenance Manual**

| | |
|---|---|
| Doc. No. | VLT-TRE-MID-15824-0264 |
| Issue | 1.5.235 |
| Date | August 23, 2007 |

| | Name | Date | Signature |
|---|---|---|---|
| Prepared | R.J. Mathar | August 23, 2007 | |
| Approved | U. Graser | | |
| Released | Ch. Leinert | | |

**Change Record**

| Issue | Date | Section/Parag. affected | Reason/Initiation/Documents/Remarks |
|---|---|---|---|
| 0.1 | 05-Mar-2002 | all | created |
| 1.3 | 21-May-2004 | 1.1 | updated URL's |
| | 21-May-2004 | Figs. 12, 13 | added |
| | 23-Jun-2004 | 3.2.2 | cftisio no longer in miopub; wcstools-3.5.3 |
| | 24-Jun-2004 | 2.7.2 | added paragraph "Recovery" |
| | 16-Jul-2004 | 2.7.3 | added description of Symptom 3 |
| | | 3.7 | removed obsolete section on `boss` FATAL ERROR |
| | | 3.5 | added |
| | 22-Jul-2004 | 3.7.2 | added |
| | 27-Jul-2004 | 3.7.3, 3.7.4 | added |
| | 17-Sep-2004 | Fig. 15, Sec 2.3, man pages | updated |
| 1.4 | 26-Sep-2004 | 1.5 | removed mio(pan)FitsByTpl |
| | 05-Oct-2004 | 3.2.2 | removed a2ps |
| | | 4 | removed miopanLogSheet |
| | | 3.6.1 | removed miopanSetup |
| | 10-Nov-2004 | all | removed 120 pages of explicit man page listings |
| | 17-Nov-2004 | 2.7.11 | extended after PPRS-00014396 |
| 1.4.236 | 23-Aug-2006 | all | transition to PDF |
| | | 5.2 | updated, extended, sorted alphabetically |
| | | 4.2.5 | added |
| | | 2.7.11 | corrected typo `bosArchiver` |
| 1.5.235 | 23-Aug-2007 | 3.7.5 | added |
| | | 5.1 | extended |

# Contents

# List of Figures

This page was intentionally left blank

# 1 USER MANUAL

## 1.1 Scope, Conformance

The Observation Software (OS) provides a layer of functionality common to the Observation Software of a generic VLT instrument: The operation preparation is supposed to be already finished and has reached a machine-readable form [12, Sec. 6] which under standard circumstances is delivered through the Broker of Observation Blocks [2, 6]. Decisions on observation modes and parameters have already been made with tools called the Observation Support Software. (Even though other parts of the software have lower levels with more flexible and extended command interfaces, all modes, including the engineering modes of frequent or non-frequent low-level laboratory tests, ought to be available through this standard "batch job" entry.)

Besides this role of an executive of observation blocks it also is the top-level door to an interactive monitoring of the subsystems, which (i) assembles their most important state parameters, including alarms that call for human attention, (ii) provides a graphical user interface that leads to their individual main panels.

Ref. [17] gives an overview of the task division between all software modules that contribute to MIDI, irrespective of whether they do have an interface with the Observation Software or not, and whether they are part of the observation preparation, the online data acquisition or the post-detection analysis.

Since the instrument user will "command" OS mediated by BOB (and, from an even higher level, by P2PP [9, 3]), the "user" would not need to gain insight into OS functionality, as long the meaning of "user" refers to an astronomer concentrating on the physical/optical aspects of the light path on one hand and the detector data fetched from the online archive on the other, skipping everything in between. This is also the "top-down" point of view taken in Fig. 1.

This wording reduces the User Manual to the following statements/chapters; these are the only aspects that might be noteworthy or peculiar from the outside.

New versions of this manual are distributed

- through the CMM module VLT-TRE-MID-15824-0264
- through the web page of the OS maintainer, `http://www.strw.leidenuniv.nl/~mathar/public/`

## 1.2 Environment

The OS is running on the Instrument Workstation `wmidi`, an HP equipped with CCSlite, integrated into the MIDI instrument LAN [15, 41]. It hosts the UNIX processes `mioControl` (the OS server which handles commands from BOB, `ccseiMsg` etc, forwards commands to subsystems and coordinates replies, and gathers portions of FITS files), and `bossArchiver` (which merges FITS files according to scripts created by the main server and announces to the online archive client when these are complete).

## 1.3 Sequentialization

The only mode of operation currently supported by the OS server is a strictly sequential operation, defined as a loop over exposures which are roughly defined by (i) receiving a series of SETUP commands, readily distributed/forwarded to the subsystems (DCS, ICS, VLTI), followed by (ii)

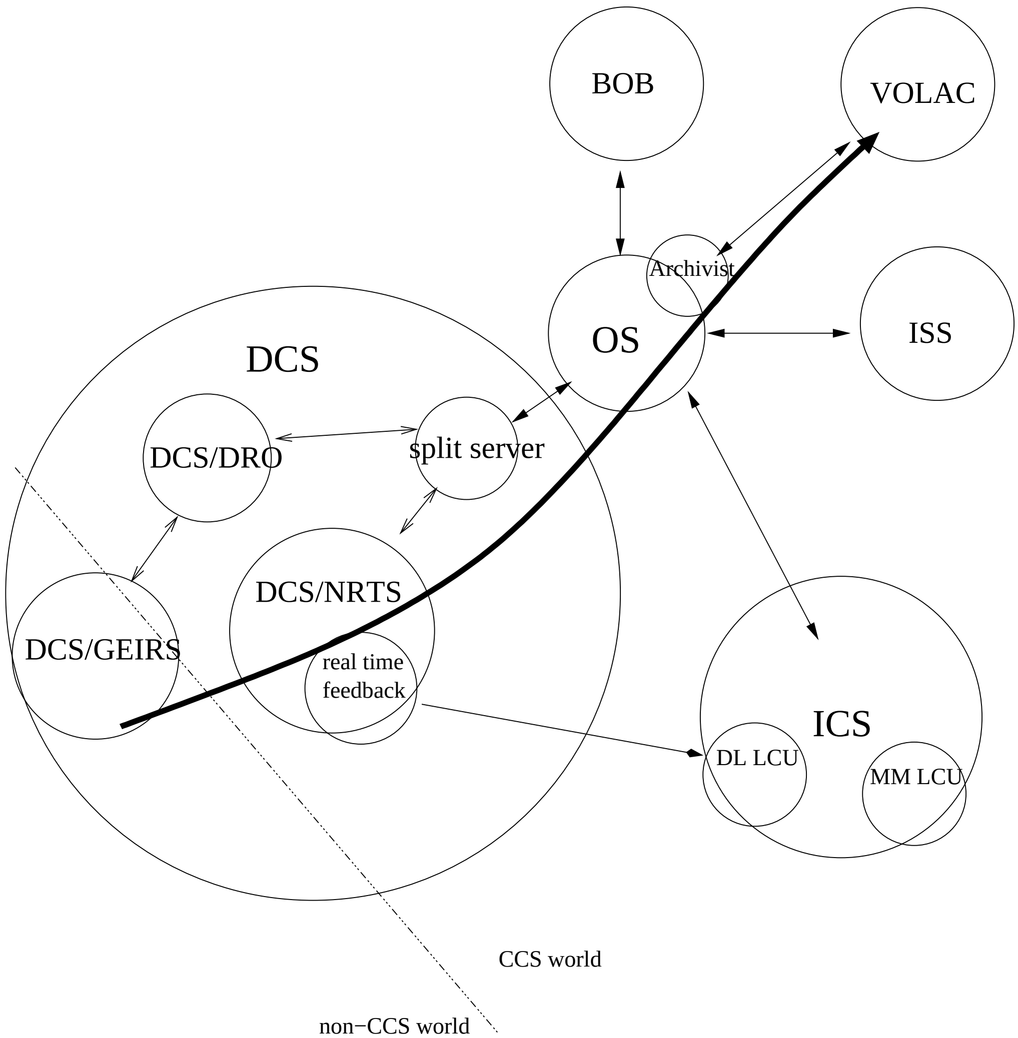



Figure 1: The relationship between MIDI program packages. The bold arrow indicates the main flow of detector data. Command exchanges follow the small arrows. Note that this diagram does not describe a map/relationship/configuration hardware ↔ software.

receiving START and WAIT, also forwarded, and (iii) finished by DCS signalling completion, which triggers (iv) completion by an archiver process. Note that this serialization means/enforces that no SETUPs are accepted while any exposure is running, but allows and supports parallelization *within* the SETUP phase since it distributes the SETUP command to the subsystems (DCS, ICS, VLTI) in parallel.

## 1.4 Time Control

Prior to the actual start of the exposures, OS generates a *schedule* that combines the information on requested integration times, requested motion of the internal delay lines ("scans") This is a rather strong control compared to operation of the underlying standard server, where usually the requirements and bandwidth of the DCS alone would set the pace of the data output after the START (i.e., once the optics is set up).

## 1.5 Data Output

### 1.5.1 VLTI FITS Format

The data files prepared for VOLAC follow the FITS table standard of [5]. Note that some tables and hierarchical keywords of the `ISS` class are not managed by OS but inherited from ISS; their interpretation is therefore out of scope of this document here and discussed elsewhere [23].

The format of the `TARTYPj` column in the `IMAGING_DATA` table is 2A, not 1A as in [5], to ensure more predictable alignment of the 16bit data on the DWS and therefore support faster data copy modes. Supposed that the detector readout is completely uncorrelated with the telescope chopping cycles, the probability of an `U` flag for the `TARTYPj` becomes $2(U+d)/c$ if chopping is active, where $U$ is the constant time window assumed for each of the two chop mirror transitions, $d$ the detector integration time, and $c$ the inverse of the chopping frequency [22].

Decoupled from any server functionality, offline display of the windowed data or conversion of this format into FITS cubes is made available through the programs `mioRtd(1)` and `mioFitsCnvrt(1)`. A request to have this display capability in the `rtd` module, `rtd(1)`, has been made in VLTSW20030093, and has been rejected by ESO.

### 1.5.2 Exposures with multiple files

The data are split such that no file becomes larger than 100 MBytes. (This maximum size is actually configurable through the `OCS.DET.FSIZMAX` keyword in the instrument configuration file `mimcfgINS.cfg`. To take effect, the `mimcfg` module must be *properly* installed after changing the keyword. Note that the configuration is only read after restarting OS [33].) The convention of [12] that `MJD-OBS` and `DATE-OBS` keywords in the primary headers characterize the start of exposures is invalid for MIDI data files. These two keywords report the start of the first frame integration of the individual file, which ensures that these two keywords are unique even if the results of an exposure are split over several files— since MIDI is a one-detector system. Correct reassembly of the individual files to exposures is possible either through `DET EXP NO`, which stays the same for all files of an exposure, or through monitoring of `TPL EXPNO` and `OCS EXPO FILENO`, or, somewhat tedious, using the `FRAME` column in the `IMAGING_DATA` table which continuously is increased until the entire exposure is completed.

Internally, a sequential file naming scheme is used; these file names are presumably the values of the `ORIGFILE` keys in the archive files. A file of the pattern `*_01.fits` is the first file of an exposure, possibly followed by a file with the same base name ending on `_02.fits`, and so on, and finally a "dummy" file ending on `_99.fits`. (The current description in http://www.eso.org/observing/dfo/quality/MIDI/basics.html is wrong.) This "dummy" file was introduced to allow explicit signalling of the BOB template termination in the case of multiple files per exposure. It contains only a primary header, and no detector data, which ensures that it does not need to be split due to size. To accomodate this case, the observation software increases `TPL NEXP` on receipt from the BOB server, before this keyword is forwarded to the subsystems and finally ends up in the primary headers.

The script `mioFitsFNames(1)` can be used to create symbolic links to the ESO archive file names in the current working directory.

### 1.5.3 Templates with Many Exposures

An index list of file names, sorted according to all exposures that result from a template run, is kept in the primary header keywords `OCS EXPOi FNAMEi`.

An estimate of the total number of files created by a template sequencer file is calculated in the sequencer file by multiple calls of `mioFits2Win(1)` and forwarded by an appropriate `SETUP` command with the keyword `OCS.TPL.NFILE` to the OS server. The OS server creates in addition the keywords `OCS TPL FILENO` in the primary headers to enumerate all data files created on behalf of the current template. Example of header keywords of a case where the template contains 2 exposures, the first creating 1 file, the second creating 2 files, and the sequencer file using `TPL.NEXP=2` and `OCS.TPL.NFILE=4` in `SETUP` commands:

**1st file** `HIERARCH ESO OCS EXPO FILENO = 1`
```
HIERARCH ESO OCS TPL NFILE = 4
HIERARCH ESO OCS TPL FILENO = 1
HIERARCH ESO TPL NEXP = 3
HIERARCH ESO TPL EXPNO = 1
```

**2nd file** `HIERARCH ESO OCS EXPO FILENO = 1`
```
HIERARCH ESO OCS TPL NFILE = 4
HIERARCH ESO OCS TPL FILENO = 2
HIERARCH ESO TPL NEXP = 3
HIERARCH ESO TPL EXPNO = 2
```

**3rd file** `HIERARCH ESO OCS EXPO FILENO = 2`
```
HIERARCH ESO OCS TPL NFILE = 4
HIERARCH ESO OCS TPL FILENO = 3
HIERARCH ESO TPL NEXP = 3
HIERARCH ESO TPL EXPNO = 2
```

**4th file** `HIERARCH ESO OCS EXPO FILENO = 1`
```
HIERARCH ESO OCS TPL NFILE = 4
HIERARCH ESO OCS TPL FILENO = 4
HIERARCH ESO TPL NEXP = 3
HIERARCH ESO TPL EXPNO = 3
```

(This is the “dummy” file with no detector data.)

Depending on the outcome of VLTSW20030219 and VLTSW20030535, this choice of primary header keywords may change in the future.

### 1.5.4 Primary Header

In exceptional cases, the keywords `RA` and `DEC` may both be reported to be zero. This is a decision made by the ISS [43, 42] system, and may for example indicate that the two telescope’s ideas of their pointing directions is too different to agree on a common value (VLTSW20020187). The MIDI OS does not attempt to appeal against this.

Note that any listings of the primary FITS header in [28] are irrelevant here, since the primary header is not forwarded by NRTS to OS (which means it is cut off when the bold line in Fig. 1 passes NRTS).

### 1.5.5 Schedules (scan presets)

The MIDI-specific `SCAN_SETUP` binary tables list on an frame-by-frame basis scheduled

- detector integration start time (MJD)

- detector integration time,
- positions of the two internal delay lines (without regard to the delay line mirror translation stage)

as scanning patterns, similar to what the IMAGING_DATA table will contain after having run the actual exposure. The important difference is, of course, that the SCAN_SETUP tables do not have any knowledge of what the fringe tracking offsets of the internal and VLTI delay lines will be in the future, and that the timing is an estimate based on the response (keyword DET.CYCLETIME) of the DCS server before the START.

### 1.5.6 Optical path differences

The LOCALOPD column in the IMAGING_DATA table and the LOC_OPL colum in the SCAN_SETUP tables refer to the rest positions of the piezos of both beams as 0 meter. (MIDI beam A is the western one, fed from the more nothern beam in the Interferometric Lab, beam 2 according to Fig. 1 of [43], and the one closer to the VINCI table in Fig. 3.5 of [44]. MIDI beam B is the eastern one, fed from the more southern beam in the Interferometric Lab, beam 1 according to [43], and closer to the Beam Compressors in Fig. 3.5 of [44].) Starting from these, voltages applied to the piezos can only reduce the optical path differences between the star and the detector; therefore the local positions are negative in general. The sign convention of the Delay Line Translation Stage (DLMT) that carries piezo A, represented by the INS DLMT keyword, is equivalent.

The optical path difference caused by the MIDI warm optics—following the VLTI sign convention [34, 3.2.6.4]— is piezo(B)−[piezo(A)+DLMT]. The value of piezo(A) is found in LOCALOPD(1), the value of piezo(B) in LOCALOPD(2). Motions of MIDI's cold M1 mirror act on both beams and therefore ought cancel when calculating the path difference.

The OPD column in the IMAGING_DATA table reflects the motion of the VLTI delay lines *relative to the blind trajectory of the VLTI geometrical model* as commanded through the reflective memory network [40, 13] via the MIDI LCU lmiics1 (and directed by NRTS commands). It does not contain measured positions, which could be obtained by reading information placed onto the reflective memory by the VLTI DL metrology system. If MIDI is not the fringe tracker, but for example FINITO, the OPD columns are all zero [24].

It should be noted that this is not an OS but rather an ICS and NRTS issue, as these data columns are compiled by NRTS based on information originating from the MIDI DL LCU.

### 1.5.7 Target table

An empty OI_TARGET table [5] is added by OS

- and if the VLTI subsystem is used in the current instrument mode (refering to OCS.MODEi.SUBSYST)
- and if the VLTI subsystem is configured with NORMAL by the OCS.TEL.ACCESS keyword.

# 2 OPERATOR MANUAL

## 2.1 CCS Environment

The OS server lives in the CCS environment `wmidi` on the Instrument Workstation (IWS) `wmidi`. This environment is started and stopped with standard VLTCS tools, explicitly with `vccEnvStart(1)` and `vccEnvStop(1)` as described in [8], or implicitly in the course of running `pkginBuild(1)` [36] or `miinsStart(1)`.

## 2.2 Startup Tool

### 2.2.1 Processes

Starting and stopping of instrument subsystems (OS, ICS, DCS, VLTI) is supported through software based on the Startup Tool [31]. The MIDI implementation [25] supports the UNIX command line calls

```
wmidi > miinsStart -proc OS
```

and

```
wmidi > miinsStop -proc OS
```

on the Instrument Workstation to start the OS server processes `mioControl` and `bossArchiver_mio` or to terminate both of them, see `miinsStart(1)`. However, these specific command lines will hardly be used, since the generic

```
wmidi > miinsStart
wmidi > miinsStop
```

already encompass them.

### 2.2.2 Graphical Interface, MIDI OS Engineering

Graphical User Interfaces are created by either following push-buttons on other Graphical User Interfaces or calling them by name. Some of them are also bound to the `ctoo` STARTUP configuration: on behalf of OS either

```
wmidi > miinsStart -panel OS_ENGINEERING
```

or

```
wmidi > miopanEngineering
```

create Fig. 2. Pressing the `STARTUP` and `SHUTDOWN` buttons of the `OS` column in this GUI is equivalent to `miinsStart` and `miinsStop` calls mentioned in the previous section. The comment of Sec 2.2.1 applies to the GUIs as well: These specific opening calls are already configured to be part of the generic

```
wmidi > miinsStart
wmidi > miinsStop
```

and only mentioned here to provide more detailed reference to what has been described in [25] from the outside.

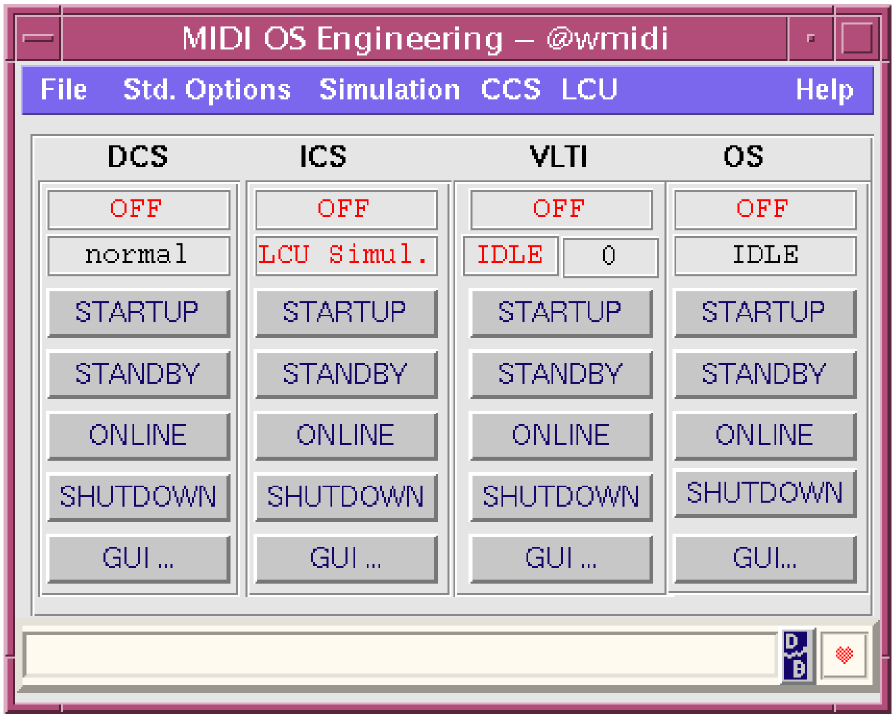



Figure 2: miopanEngineering—derived from the template instrument standard `xxopanEngineering` [18]— is also accessible by the `Engineering` option on the top bar of Fig. 5.

## 2.3 Command-line Startup and Shutdown

The aformentioned Startup Tool starts the servers in a form equivalent to the command line calls

```
wmidi > mioControl
wmidi > bossArchiver MIDI
```

Higher log levels are available via command line options listed in `mioControl(1)` and in `bossArchiver(1)`, or by changing the `OCS.CON.LOGLEVEL` in `mimcfgSTART.cfg`. The activation of the `bossArchiver` might change according to VLTSW20040257.

## 2.4 Monitoring Graphical User Interfaces

Besides Fig. 2, the `miopan` module offers the following MIDI specific graphical interfaces:

### 2.4.1 Distributor

The call

```
wmidi > miopan
```

creates the GUI of Fig. 3.

Its main purpose is to provide a fast shortcut to other functionality through its `OS` menu button without need to remember the names of other interfaces by heart. The two lowermost buttons provide means to look at the raw FITS [10, 26] files produced by the instrument on the instrument workstation in the format specified in [5]. (Here, “raw” means on an exposure-by-exposure basis with no inherent support of data analysis.) This is generally done by an intermediate representation

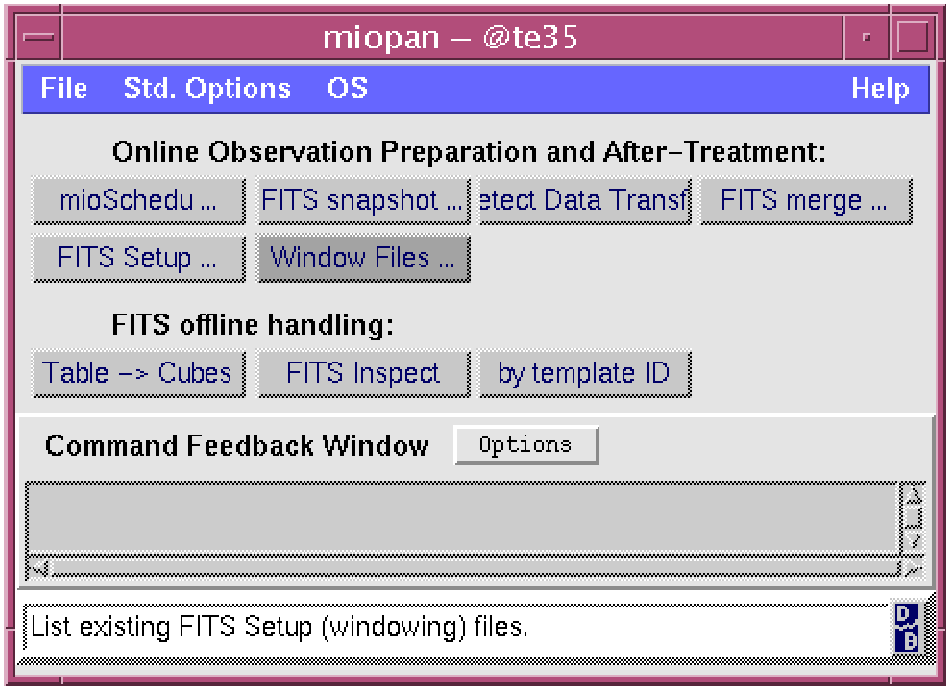



Figure 3: miopan - pops up if the `GUI...` button in the `OS` column of the GUI in Fig. 2 is pressed.

of the `DATAj` columns of the `IMAGING_DATA` table as FITS cubes, for which image displayers already exist.

### 2.4.2 MIDI Control

Fig. 5 is called by

```
wmidi > miopanControl
```

or

```
wmidi > miinsStart -panel OS_CONTROL
```

The `ICS` submenu of the blue top bar produces ASCII outputs of sensor values and motor positions collected by ICS with examples shown in Figs. 9, 10 and 11. These outputs are not alternatives to the `mimpan` output [27] but complementary, concise, one-shot composites of the most important information in the format of `miiInspath(1)`.

Its lower area displays a selective log monitor on the IWS, hence displaying the operational logs of OS, ICS (including the LCUs) and DCS/DRO. All these log lines are obtained by filtering the usual `logFile` with `mioOpLogs(1)`, the main difference towards the `logMonitor(1)` [39, §2.11] being that the focus here is exclusively on the operational logs.

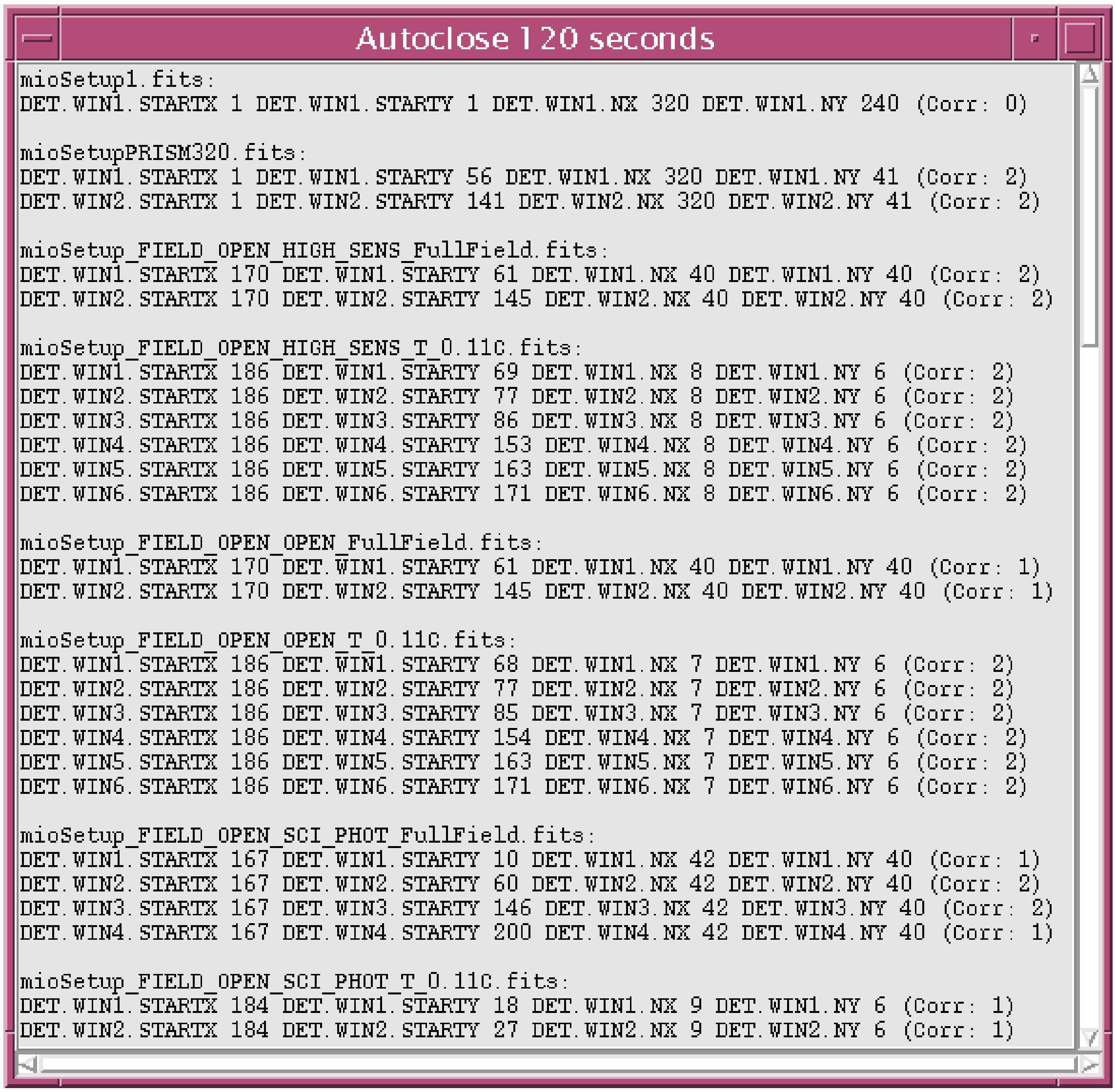



Figure 4: This list of FITS template files is created if the `Window Files ...` button of the GUI in Fig. 3 is pressed.

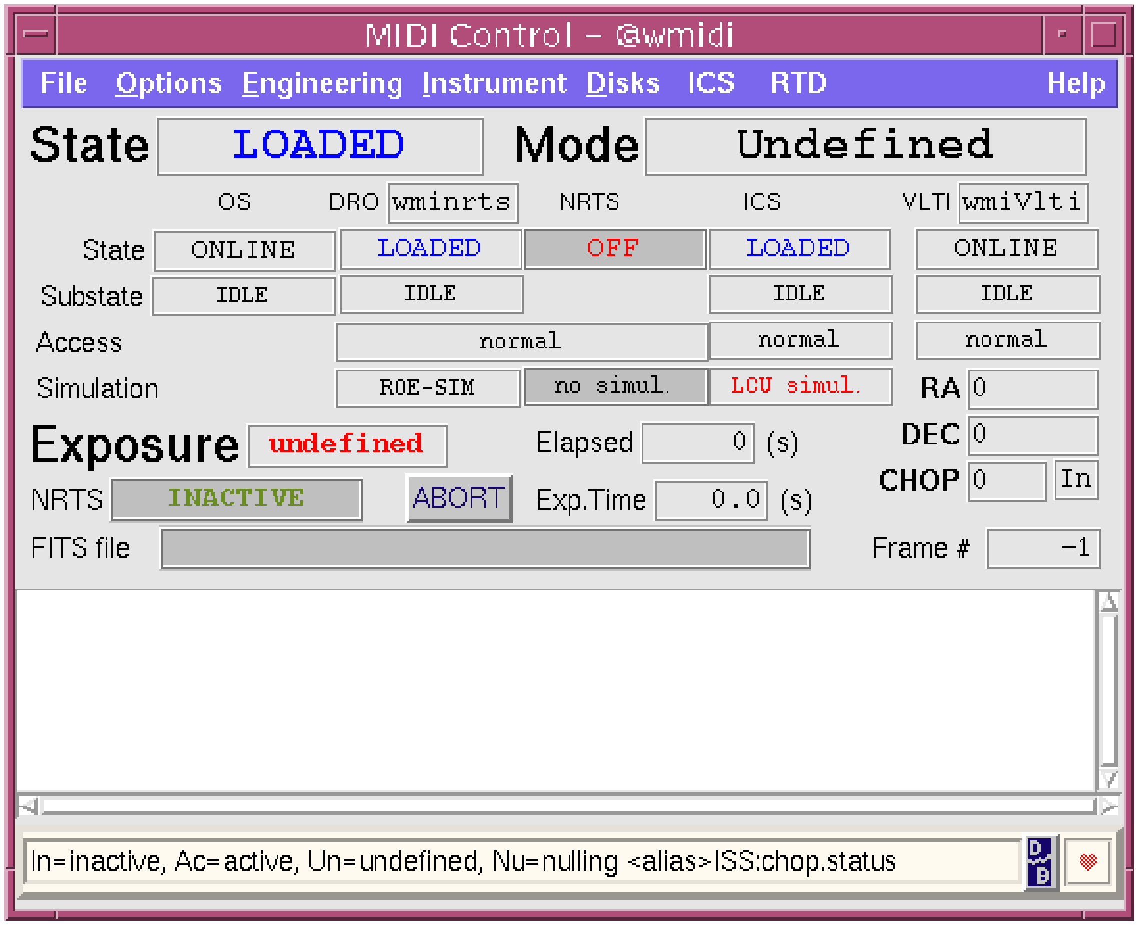



Figure 5: miopanControl—derived from the template instrument standard `xxopanControl` of [18]—is explained in `miopanControl(1)`.

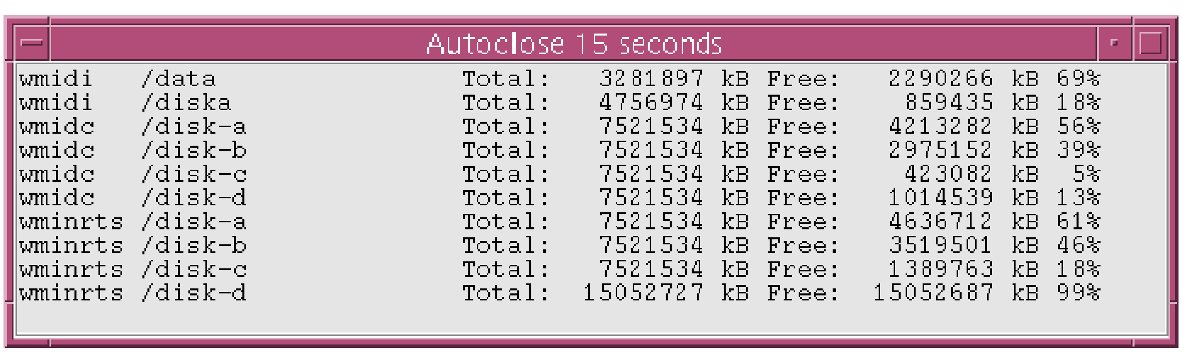
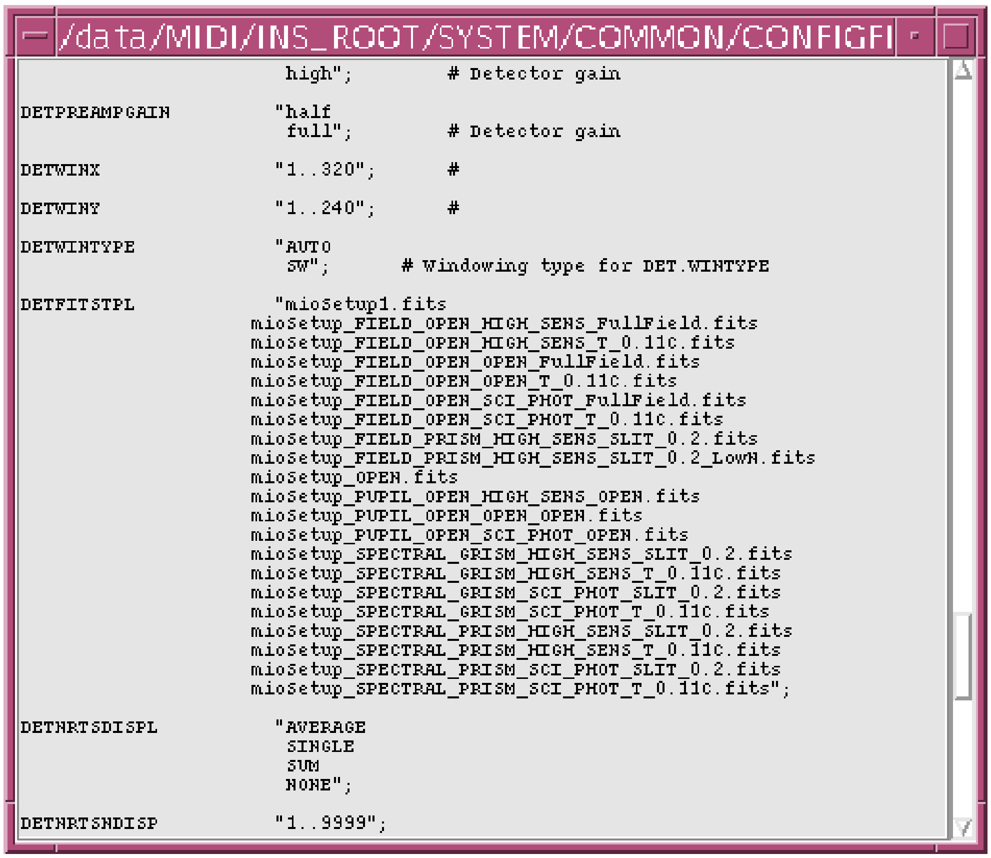



Figure 6: This auxiliary window is obtained by pressing the `Disks` button in the header bar of Fig. 5. It is based on `mioIstDiskInfo(1)`.



/data/MIDI/INS_ROOT/SYSTEM/COMMON/CONFIGFI

```
                         high";        # Detector gain

DETPREAMPGAIN           "half
                         full";        # Detector gain

DETWINX                 "1..320";      #

DETWINY                 "1..240";      #

DETWINTYPE              "AUTO
                         SW";      # Windowing type for DET.WINTYPE

DETFITSTPL              "mioSetup1.fits
                      mioSetup_FIELD_OPEN_HIGH_SENS_FullField.fits
                      mioSetup_FIELD_OPEN_HIGH_SENS_T_0.11c.fits
                      mioSetup_FIELD_OPEN_OPEN_FullField.fits
                      mioSetup_FIELD_OPEN_OPEN_T_0.11c.fits
                      mioSetup_FIELD_OPEN_SCI_PHOT_FullField.fits
                      mioSetup_FIELD_OPEN_SCI_PHOT_T_0.11c.fits
                      mioSetup_FIELD_PRISM_HIGH_SENS_SLIT_0.2.fits
                      mioSetup_FIELD_PRISM_HIGH_SENS_SLIT_0.2_LowN.fits
                      mioSetup_OPEN.fits
                      mioSetup_PUPIL_OPEN_HIGH_SENS_OPEN.fits
                      mioSetup_PUPIL_OPEN_OPEN_OPEN.fits
                      mioSetup_PUPIL_OPEN_SCI_PHOT_OPEN.fits
                      mioSetup_SPECTRAL_GRISM_HIGH_SENS_SLIT_0.2.fits
                      mioSetup_SPECTRAL_GRISM_HIGH_SENS_T_0.11c.fits
                      mioSetup_SPECTRAL_GRISM_SCI_PHOT_SLIT_0.2.fits
                      mioSetup_SPECTRAL_GRISM_SCI_PHOT_T_0.11c.fits
                      mioSetup_SPECTRAL_PRISM_HIGH_SENS_SLIT_0.2.fits
                      mioSetup_SPECTRAL_PRISM_HIGH_SENS_T_0.11c.fits
                      mioSetup_SPECTRAL_PRISM_SCI_PHOT_SLIT_0.2.fits
                      mioSetup_SPECTRAL_PRISM_SCI_PHOT_T_0.11c.fits";

DETNRTSDISPL            "AVERAGE
                         SINGLE
                         SUM
                         NONE";

DETNRTSNDISP            "1..9999";
```

Figure 7: This auxiliary window shows the Instrument Summary File (ISF). It is obtained by pressing the **Help** button in the header bar of Fig. 5, selecting the **ISF display** option, and chosing the file **MIDI.isf**.

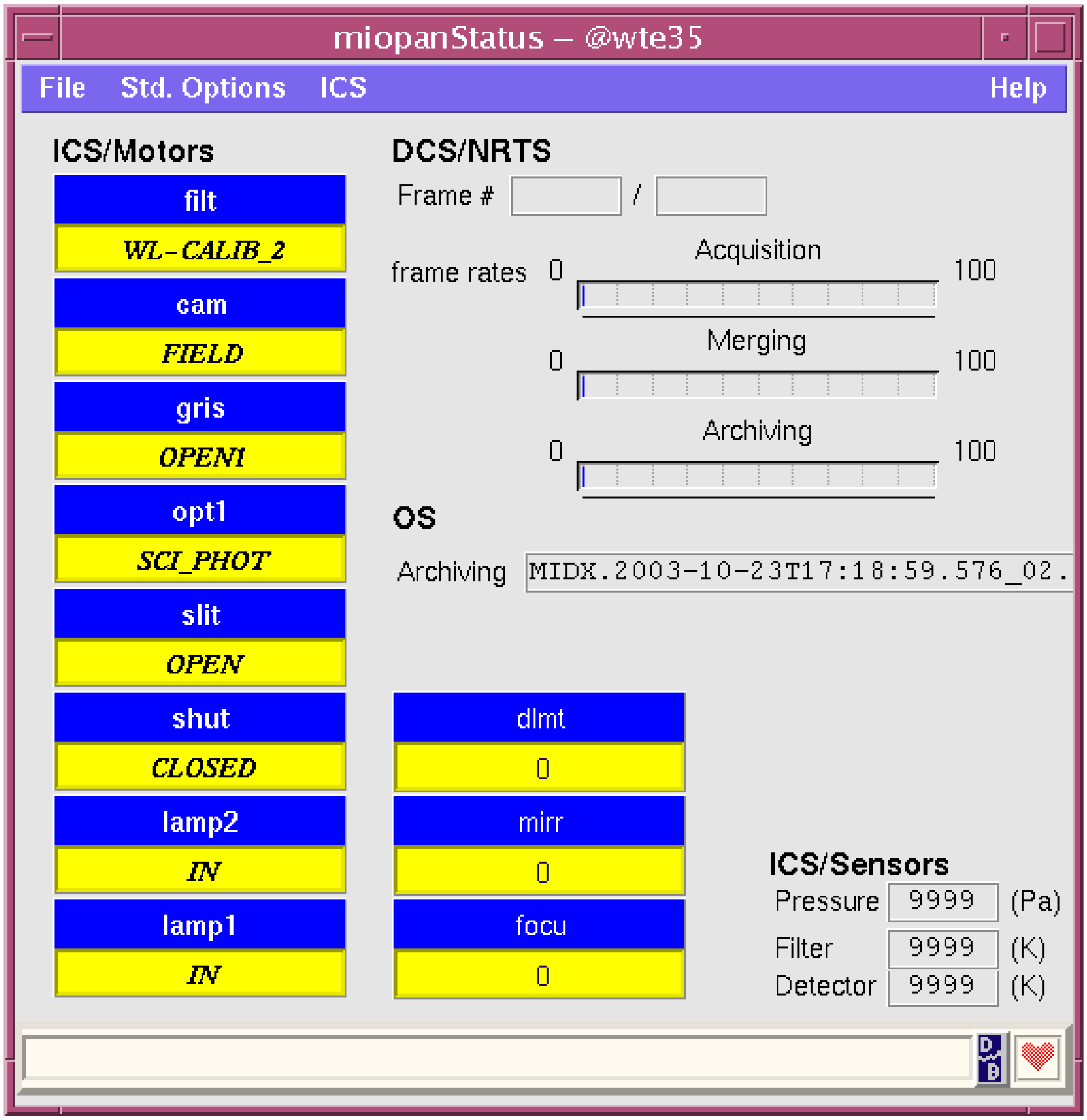



Figure 8: miopanStatus, see `miopanStatus(1)`

### 2.4.3 MIDI Status

Fig. 8 is called by

```
wmidi > miopanStatus
```

or

```
wmidi > miinsStart -panel OS_STATUS
```

## 2.5 Server Command Interface

### 2.5.1 Access

The command definition table implements the "standard" list of commands inherited from the underlying BOSS server [33],

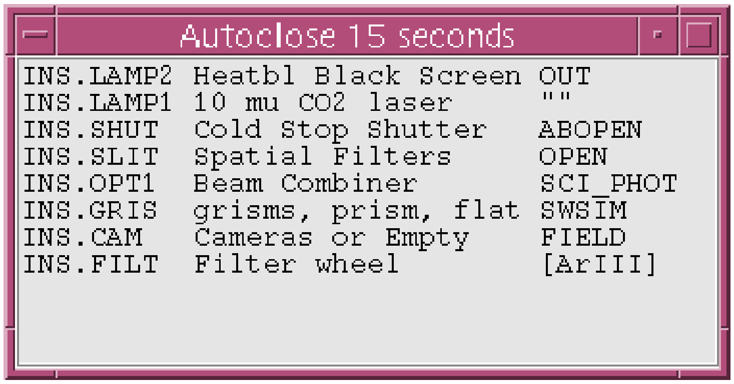



Figure 9: This auxiliary window is obtained by pressing the `ICS` button in the header bar of Fig. 8 and selecting the `Light Path` option.

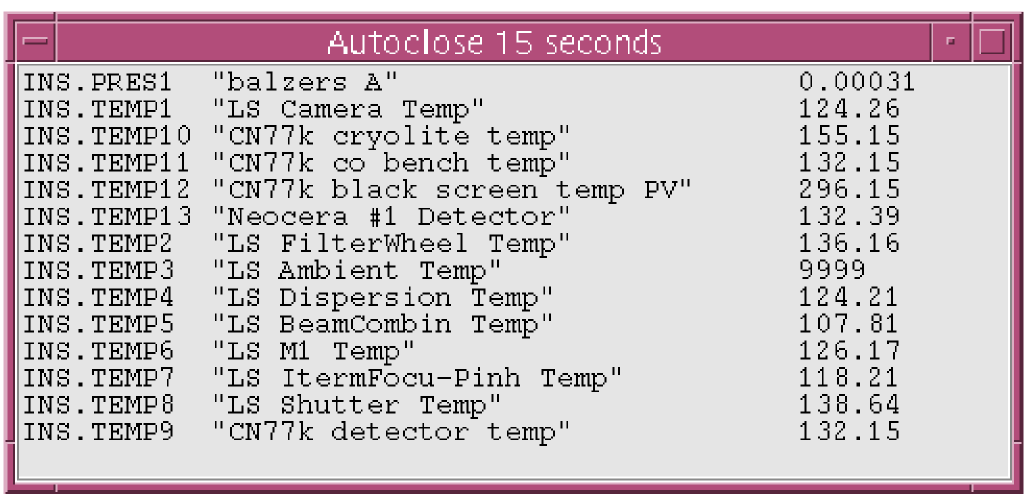



Figure 10: This auxiliary window is obtained by pressing the `ICS` button in the header bar of Fig. 8 and selecting the `Temperatures` option.

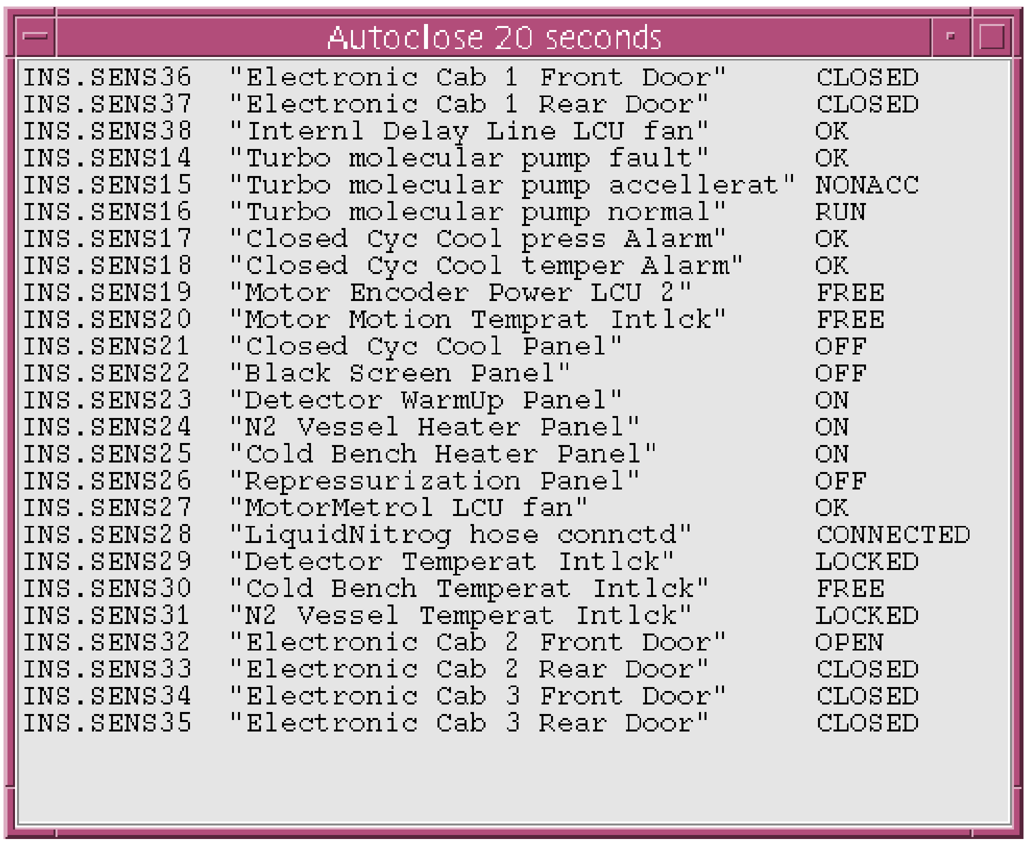



Figure 11: This auxiliary window is obtained by pressing the `ICS` button in the header bar of Fig. 8 and selecting the `Interlocks` option.

- plus a large subset of commands known to the issifControl process [43]. Making use of the forwarding capabilities, configured through the OCS.TEL.KEYWFILT keywords of mimcfgINS.cfg, the latter means that
  - NRTS may steer the telescopes by sending issifControl commands to mioControl
  - BOB may send all issifControl commands to mioControl—this is not simple, however, since the name of the control process is hardcoded in the private data section in the isstclISS.tcl source code of the isstcl module at the time of this writing.
- minus the pair of commands PAUSE and CONT that do not make sense for the instrument, as no means for synchronzied suspension of the data flow are implemented in the servers shown in Fig. 1.

From an alternative point of view, this includes the "mandatory" commands of workstations, STATUS, STATE, STANDBY, ONLINE, OFF and VERSION of [14, §3.1.4.3]— with the exception (negative list) of INIT, SELFTEST, SIMULAT, STOPSIM, STOP and VERBOSE— and includes ABORT, COMMENT, CONT, END, PAUSE, START and SETUP of [19].

The commands may originate from BOB, from the UNIX shell via msgSend(1), from panels via panSendCommand(3), from the sequencer shell [1], or from any other process booking into CCS. Shortcuts to the STANDBY and ONLINE commands are provided by two buttons in the OS column of Fig. 2, and a shortcut to the ABORT command in Fig. 5. It is generally advisable to use the CCS Engineering User Interface

```
wmidi > ccseiMsg &
```

documented in [39, §2.18.4] to send commands to OS, since the full and up-to-date Command Definition Table (CDT) will be listed in the Command selector of the GUI. (Select wmidi as the Environment and mioControl as the Process.)

### 2.5.2 Standards

The OS server inherits a full set of standards of operation from the base class server [33], like

- Forwarding STANDBY and ONLINE commands to the subsystems. (The exception is that state changing commands like these are not forwarded to the VLTI.) Sending STANDBY to the subsystems when starting or exiting (and comes as a surprise for novices). This may be seen as providing "master control" and implementing "state alignment."
- Automated creation of "merger" scripts for the archiver in conjunction of synchronized requests to the ICS and ISS subsystems to generate binary FITS tables.

In all cases of interaction with the telescope(s), the MIDI OS has replaced the "TCS" subsystem by the "VLTI/ISS" subsystem as embodied by the bossVLTI module wherever applicable.

The at parameter of the START command is effectively ignored, and the actual exposure start time in the schedules (§1.4) is put about ten seconds into the future from the time when mioControl receives the START command. If mirror chopping is active at that time, the start of the first frame integration is aligned to the begin of a chopping (half) cycle on the target, otherwise it is simply set to start at a straight second.

### 2.5.3 Example

The probably fastest (testing, BOB-independent) way of running an exposure that generates an output file is to stay with any defaults already used by the subsystems, i.e., to use a minimum set of SETUP commands necessary to request/select an exposure ID to "trigger" generation of an output file in the `$ARCDATA` directory:

```
wmidi > msgSend "" mioControl online ""
wmidi > msgSend "" mioControl clean ""
wmidi > msgSend "" mioControl setup "-expoId 0 -function INS.MODE AUTOTEST"
wmidi > msgSend "" mioControl start "-expoId 1"
wmidi > msgSend "" mioControl wait "-expoId 1"
```

(See also the shell script `mio/test/mioTst`. This also takes into account that NRTS [25] demands a minimum set of keywords to be specified.) Use of an instrumen mode that involves the VLTI subsystem (e.g. `STARINTF`) instead of `AUTOTEST` would also merge the status information of the `issifControl` process, if the TCS/VLTI subsystem was flagged to be available at `mioControl` start time.

## 2.6 Support Programs

Programs of the `mio` module beyond the core functionality described above are listed below.

### 2.6.1 Result File Copies

The default path of the archival of data created in the `DHS_DATA` directory by the `bossArchiver` leads to the online archive, see `start-vcsolac(1)`. Additional online copies of these FITS files to another workstation may be forked with `mioFtp2Ews(1)`, if the `mio/mio/test` section of the `mio` module has been compiled.

### 2.6.2 Detector Data Frame Display

`mioRtd(1)` and Fig. 12) is based on `rtd(1)` [7] and displays detector data stored in the VLTI FITS file format on a frame-by-frame basis and as films.

### 2.6.3 Online Archival

The online archival programme, which transfers data files that have been merged by OS, is started with `start-vcsolac`

- at IWS boot time by an entry in `/sbin/rc3.d`—depending on some definitions of `bash` environment variables—
- at OS startup time from within `mioControl(1)` if not yet running.

Blocking archival of any data that are regularly taken by OS is intended to be difficult, but may be worth while if large test files are created that are known to be junk. The procedure in this case would be

- start OS as usual (Sec. 2.2.1)

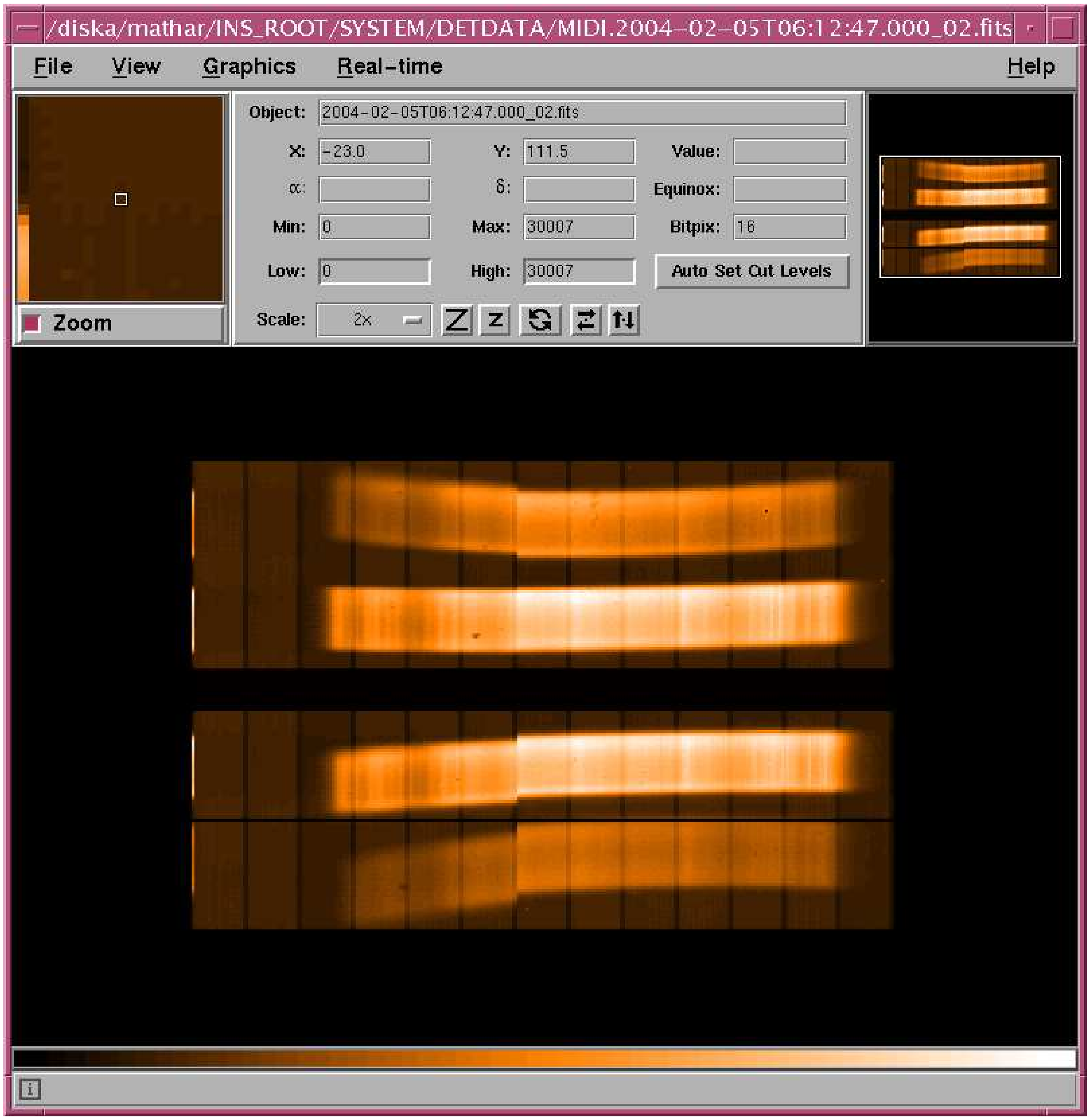



Figure 12: mioRtd - provides instances of the RTD to display VLTI FITS files. Could also be called by pressing the `RTD` button in the menu of Fig. 5.

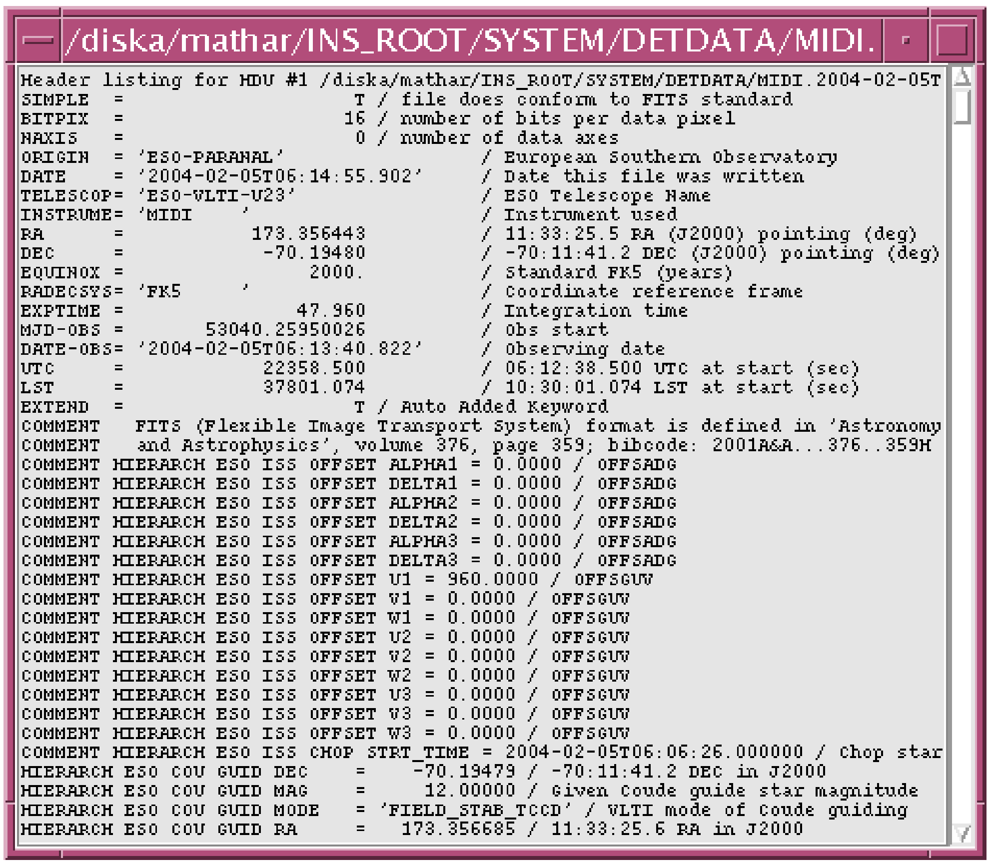



Figure 13: This auxiliary ASCII listing is opened together with Fig. 12 if `mioRtd` has been called from Fig. 5. It combines the outputs of `mioFitsListstruc(1)` and `mioFitsListhead(1)`.

- explicitly stop the client process with `cleanup-vcsolac`
- collect data with the intent to discard them
- remove the archiver's notion on files waiting to be archived with `rm $DHS_DATA/*.fits`. This is the crucial step since otherwise restart of `vcsolac` would resume handling these.
- re-start OS as usual (Sec. 2.2.1) to start another session that will resume standard archival of the FITS files.

## 2.7 FAQ and Troubleshooting

### 2.7.1 Frozen VLTI Substate

**Symptom** The VLTI substate indicated in some panels is frozen and remains `unknown.`

**Cause** The driver of the `vltisim` package does not implement state changes of the `<alias>ISS:vltiState.vltiSubState` database attribute—actually what is the source in `@$VLTI_ENVNAME:Appl_data:VLTI:issmsw.substate` before scanning.

**Workaround** Ignore and/or write a SPR requesting improvement of the `vltisim` CMM module.

### 2.7.2 Persistent OS substate "Observing"

**Symptom** OS does not return to the `online/idle` state after starting an exposure but remains in the `online/observing` substate.

**Cause** NRTS does not drive its exposure status database point of `OCS.DET1.DBEXPSTS`. The individual reasons might be the process `minrtsControl` died on `wminrts` or the scanning to `wmidi` fails.

The `Abort` button of BOB only terminates its internal handling of commands, but does not forward an `ABORT` command to the OS server [20, §5.5].

**Workaround** Check with `vccEnvCheck(1)` that `wminrts` is alive. Check with `msgSend -n wminrts minrtsControl PING ""` that the NRTS server is still alive, and check with `scanei(1)` that `wminrts` is scanned. Note that starting the DCS system on `wmidi` with `miinsStart` activates the scanning, whereas manual start of NRTS on `wminrts` with `minrtsStart` does not.

**Recovery** Restart OS as usual, either by

```
wmidi > miinsStart -proc OS -restart
```

or pressing the `SHUTDOWN`, then `STARTUP` button in Fig. 2.

### 2.7.3 Persistent OS state "off"

**Symptom 1** OS does not reach the `online/idle` state at startup.

**Cause** One or more of the subsystems configured as `available` do not go `online`. State alignment between OS and the subsystems means that the OS server takes the instrument state as the "lowest common multiple" of the subsystem states.

**Workaround** Use `miopanControl` (Fig. 5) to figure out which of the subsystems planned to have `normal` access did not move up to a sufficiently high state. Notice that the state of the OS server and the "aligned" instrument state are two different things.

**Symptom 2** The UNIX process `mioControl` does not start but produces `permission denied` errors in `/vltdata/tmp/logFile`.

**Cause** The attempt to copy `mioSetup*.fits` files with `mioSetupDistr(1)` from `wmidi` over to `wminrts` (see Sec. 3.6.1) failed due to insufficient file write permissions for the `midi` account on `wminrts:$INS_ROOT/SYSTEM/MISC`.

**Workaround** Login as `midimgr` on `wminrts` and use

```
wminrts > chmod g+w $INS_ROOT/SYSTEM/MISC/mioSetup*.fits
```

On the long term ensure that the corresponding `Makefile` of the CMM module `minrts` grants sufficient rights if it compiles the `INS_ROOT` directory from scratch under the `midimgr` account.

**Symptom 3** The UNIX process `mioControl` does not start but complaints about missing reference files in `/vltdata/tmp/logFile`.

**Cause** The reference files provided by the `mioseq` module are either incomplete if compared with the `OCS.MODIi.SETUP` list in `mimcfgINS.cfg`, or faulty.

**Workaround** Ensure that the `mioseq/config/*.ref` in the `mioseq` module are correct, and that the `make clean all man install` step in the `mioseq` module distributes them correctly.

### 2.7.4 Cannot complete User defined function

**Symptom** `mioControl` returns with the error "bossExpoCtrl.C:1832 1090 922 1 S bossERR_CMD_FAILED_APP : Could not complete User-Defined function during the execution of command START."

**Cause** One of the MIDI specific tasks of the OS server was not completed sucessfully after receiving the `START` command. The prospective causes are

- The contents of the `SCHEDU` command was rejected by `lmiics1` because the motion schedules for the internal delay line was out of bounds.
- The contents of the `SCHEDU` could not be delivered to NRTS.
- The environment was (partially) started as `midimgr`, ignoring the explicit advice of [25], which means that data file permissions in `$INS_ROOT/$INS_USER/DETDATA` are insufficient.

**Workaround**

- Check that the internal delay line parameters stay both in the range of about 0 to $-150 \cdot 10^{-6}$ for the entire duration of the schedule. Do not forget the exponent (units are meter). The `Ascii dump` of Fig. 14 may help to debug.
- Check that the environment `wminrts` and the main server `minrtsControl` are still alive (`vccEnvCheck(1)`, `miopanControl(1)`)

### 2.7.5 Complaint on missing HEADER file

**Symptom** The OS server `mioControl` does not start but issues errors like "Could not locate file "HEADER" according to environment variable """

**Cause** Reference files specified in `mimcfgINS.cfg` with the keywords `OCS.MODEi.SETUP` are missing in `$INS_ROOT/SYSTEM/COMMON/SETUPFILES/REF` due to incomplete installation of `mioseq`.

**Workaround** Login as `midimgr` on `wmidi` and call `make all man install` in `$PKGIN_BUILD/../miotsf/src` and then in `$PKGIN_BUILD/../mioseq/src` to recreate them.

### 2.7.6 Logline of undeclared file name

**Symptom** The `logFile` contains lines of the format "Filename is not declared for exposure ... on det. DCS"

**Cause** This is a log line created by the BOSS base class—which is inherited by the `mioServer`—after `START`.

**Workaround** This may be ignored, since the MIDI file names are declared later, when the schedules have been created and when the time stamps of the first exposure are available to become part of the file names.

### 2.7.7 Logline of subsystem being not on the list

**Symptom** The `logFile` contains lines of the format "WARNING !! Setting subsystem ... that is not on the list..."

**Cause** `SETUP` keywords have been sent which are not compatible with the most recently set `INS.MODE` according to the instrument modes declared through the `OCS.MODEi` configuration keywords. The most likely reason is that `TEL` keywords were used in combination with the `AUTOTEST` mode.

**Workaround** Correct the sequencer or shell scripts that set up and run the exposure.

### 2.7.8 Stopped Graphical User Interfaces

**Symptom** Starting a GUI in the background (by use of the UNIX shell ampersand) freezes the contents of this GUI, which includes that refreshing of the window is missing. The issueing shell reports the corresponding background job as "stopped."

**Cause** The `bash(1)` under HP-UX works with a serial line interface policy that lets the background process of the GUI compete with the foreground process on the association/binding to the serial line interface.

**Workaround** Use one of the following:

1. Issue the command

   ```
   set +m
   ```

   on the HP-UX `bash(1)` before pushing GUI's into the background.

2. Redirect the standard input and (error) outputs of all `exec` commands in `tcl` scripts to `/dev/null` as far as possible. Edit the default line of the executable in the same way to become

   ```
   exec seqWish "$0" ${1+"$@"} >/dev/null 2>/dev/null </dev/null
   ```

   This is an option only available to the developpers.

3. Do not use HP-UX `xterm`, for example by using `rlogin` from a computer running another shell or SunOS.

4. Run the GUI's with their own private `xterm(1)` following the template

   ```
   wmidi > xterm -geometry 80x2 -e xxmyGui &
   ```

Do not push GUI's into the background with CTRL-`Z` (or whatever the suspend key strokes according to `stty(1)` are). If necessary, use the `fg` command of the `bash(1)` to revive.

#### 2.7.9 Strange and unrealistic temperature and pressure recordings

**Symptom** The OS Control GUI reports strange temperatures or pressures like 8888.0 or 99999.0.

**Cause** ICS uses unlikely values of temperatures and pressures to indicate that sensor devices are either used in "hardware simulation" or cannot properly exchange commands with the actual electronic hardware.

**Workaround** This is not an OS but an ICS issue, see [21, 27].

#### 2.7.10 Detector Data Disk Full

**Symptom** The START command returns an error reporting that the disk space is insufficient for the scheduled exposure. The call

```
wmidi > mioIstDiskInfo $DETDATA
```

or clicking on the `Disks` menu button of Fig. 5 confirms this.

**Cause** The IWS disk is of finite size, and the data files accumulated in `$INS_ROOT/SYSTEM/DETDATA` do not leave enough space; the requirement is estimated from the number of detector integrations and the total pixel area of the FITS template files, see eg `mioFits2Win(1)`.

**Workaround** First check that the keyword DET.NDIT is of the correct order of magnitude to avoid over-estimation of the requested space. (The estimated disk requirement is roughly proportional to this value.) Decide whether a backup disk exists with sufficient capacity to swallow some of the detector data files, or whether detector data files must be removed from the IWS disk (in case of which one relies on OLAC as the backup system). Calculate a list of "older" data files and (re)move them:

```
wmidi > cd $INS_ROOT/SYSTEM/DETDATA
wmidi > export t1=`ls -1 MIDI*.fits | wc -l`
wmidi > export t2=`dc -l <<++
wmidi > $t1 3 / f
wmidi > ++`
wmidi > export t1=`ls -1r MIDI*.fits | tail -$t2`
```

followed by either

```
wmidi> mv $t1 /diskc/04/midi/
```

if this backup directory exists under `/diskc`, for example, or by

```
wmidi> rm $t1
```

if the files are to be deleted.

This procedure might be repeated during a night.

#### 2.7.11 No Data Files on `wvgoff`

**Symptom** The MIDI FITS files that are available on `wmidi:$INS_ROOT/SYSTEM/DETDATA` are not forwarded to the offline machine, `wvgoff:/data/raw`.

**Cause 1** The corresponding disk on `wvgoff` is full as confirmed with

```
wvgoff> df -k /data/raw
```

**Workaround** Remove superfluous files on the disk, and try to restart the process that subscribes to the online archive [35]

```
wvgoff> dhsSubscribe
```

**Cause 2** The process `bossServer` is not running on `wmidi`, which means that files with suffixes like `hdr.det`, `btbl.fits` and `.arf` are accumulating in `wmidi:$INS_ROOT/SYSTEM/DETDATA`. Also, the command

```
wmidi> msgSend "" bossArchiver_mio PING ""
```

says that it cannot get information on the process, and for each FITS file generated an error window with the same message pops up (unless the shell variable `NO_ERR_DISPLAY` has been set).

**Recovery** Restart the OS with a

```
wmidi> miinsStop -proc OS
wmidi> miinsStart -proc OS
```

pair, or even simpler, restart only the archiver with

```
wmidi> msgSend "" bossArchiver_mio EXIT ""
wmidi> bossArchiver MIDI
```

If this does not work, start `mioFtp2Ews(1)` to build another mean of forwarding the data from `wmidi` to `wvgoff`, but note that these do not trigger the online pipeline handling (standard archival).
`mioArchiv(1)` is an auxiliary program to merge the FITS pieces that have been left over in `wmidi:$INS_ROOT/SYSTE`
in this case.

# 3 MAINTENANCE MANUAL

The names of the Observation Software modules in ESO's Configuration Management Module are `mio`, `miopan`, `miopub`, `mioseq`, and `miotsf`. The contents of the first three of these is covered here, whereas `mioseq` and `miotsf` are described elsewhere [25, 4].

## 3.1 Installation

### 3.1.1 Instrument Workstation

The standard installation of the OS, ICS and the DRO part of the DCS software on the IWS uses one of various options of `pkginBuild` [36] to retrieve and compile the source codes as layed out in the files `wmidi:$PKGIN_BUILD/miins/config/miinsINSTALL*.cfg` and `wminrts:$PKGIN_BUILD/minrts/config/minrtsINSTALL*.cfg`. A 3-line entry of the format

```
INSTALL.MODULEi.NAME "miopub";
INSTALL.MODULEi.NAME "mio";
INSTALL.MODULEi.NAME "miopan";
```

in the pkgin configuration file suffices for compilation. Without use of `pkginBuild`, this is equivalent to

```
wmidi > cd $PKGIN_BUILD/../miopub/src
wmidi > make clean all man install
wmidi > cd $PKGIN_BUILD/../mio/src
wmidi > make clean all man install
wmidi > cd $PKGIN_BUILD/../miopan/src
wmidi > make clean all man install
```

The `midi/.netrc` file is needed by `mioSetupDistr(1)` and `mioFtp2Ews(1)`. As it contains verbatim passwords, it is not stored in CMM, but must be edited manually. The contents is
`machine wminrts login midi` *passwd*
`machine wmidc login midi` *passwd*
The second line is not needed if DCS is run in the newer single machine configuration [28].

### 3.1.2 Preprocessor Workstation

The equivalent standard installation of the NRTS software on the PP WS [25] ensures that static configuration files for schedules and for FITS templates are consistent. The current version of the Makefile compiles only a small part of the module on the PP WS `minrts`, and leaves most of it out using a switch that senses the type of operating system (SunOs). This reduces compilation times on the PP WS.

## 3.2 CMM modules

### 3.2.1 mio

The directory `mio` contains the core part of the master server `mioControl` in form of classes derived from the `bossSERVER` and the `bossINTERFACE_DCS` classes. In addition, the class to represent piezo schedules is kept here, the windowing template files (Sect. 3.6.1), and auxiliary programs (Sect. 2.6).

The directory mioDxf keeps the file transfer module, which has been derived from the dxf module [32] with enhanced functionality concerning (i) disk space checks (ii) synchronization with symbolic links, (iii) optional removal of files on the source machine after completion, (iv) optional flagging of completion by filename dump into a OLDB attribute. See the discussion under VLTSW20010324.

### 3.2.2 miopub

This directory contains "third party" software packages not written specifically on behalf of MIDI. Search for the string mathar in the unzipped source code to look for changes that have been made to adapt these packages to the VLT programming environment.

1. CCfits is a C++ class library on top of cfitsio used at various places within the mio module to create or edit binary FITS files, see http://heasarc.gsfc.nasa.gov/docs/software/fitsio/CCfits/.
2. wcstools-3.5.3 is D. Mink's library from http://tdc-www.harvard.edu/software/wcstools. It is needed to correct the CMM module cat for use in mioFitsFNames(1). Since the associated SPR VLTSW20030475 has been closed by ESO without action, it looks like this fix of the ESO module bug must stay forever.

### 3.2.3 miopan

Contains the (i)tcl codes to create Figs. 2–8 and the java code to create Fig. 15.

## 3.3 Configuration

The configuration through the files mimcfgINS.cfg and mimcfgSTART.cfg contained in the CMM module mimcfg is described in [33]. The subsystems shown in Fig. 1 are represented by one server each:

| Subsystem | Category | standard server | alternative (simulating) servers |
|---|---|---|---|
| ICS | OCS.INS1 | miiControl@wmidi | |
| DCS | OCS.DET1 | midControl@wmidi | |
| VLTI | OCS.TEL | issifControl@wvgvlti | issifControl@wmiVlti |

From the point of view of OS, there is only one detector subsystem. (This is automatically achieved by installing a single DCS representative in mioSERVER::SubsystemInterfaces().) The OCS.DET[23] keywords are usually ignored; hence OCS.DET.NUM could be set to 1. The Startup Tool (miinsStoo(3), [31]) and the split server midControl, however, use OCS.DET2 to configure DCS/DRO and OCS.DET3 to configure DCS/NRTS [28]. Therefore, at the time of this writing, OCS.DET.NUM must be kept at 3.

If the ISS/VLTI is not used, specifically if

- light sources represented by ISS (telescopes, siderostats, ARAL) are not used, but only sources internal to MIDI (the $CO_2$ laser, the black screen,...)
- no communication with the VLTI server or its simulating counterpart is necessary

the OCS.TEL.AVAILABLE keyword could be set to IGNORE which means that

- ISS/VLTI related FITS tables are not generated

- the state of the VLTI subsystem does not play a role while going `ONLINE`

Since the DCS module is not yet doing this, some support of running DCS simulations is also built into OS when it receives the `ONLINE` command: Additional SETUP commands are forwarded to DCS as documented in the section about the `OnlinePostProc` member function in `mioSERVER(4)`.

### 3.4 Piezo motion and exposure schedules

In this chapter, a schedule is a piece of information on

- when a series of detector integrations should start,
- how long the individual integrations should last,
- where the two piezos representing the internal delay line ought be positioned for the duration of this integration,
- which residual time should be allocated for background tasks (data readout, piezo mirror repositioning) to complete such a cycle, and
- how often these atomistic operations ought be repeated/concatenated.

Adorned by some administrative information (a name, or interspersed markers that may be interpreted by schedule readers, for example), this ends up in the strict layout described in the DATA FORMATS section of `mioSchedCrea(1)`.

The most compact, binary representation of this information is distributed by OS in form of CCS `SCHEDU` messages to the subsystems. The building blocks and an entire schedule are represented by the C++ classes in `mioSchedCREA.C`, which are given a command line interface in two more or less equivalent ways:

- in form of the ASCII editor `mioSchedCrea(1)`
- in form of the `mioSchedu` GUI of Fig. 14.

Both ways allow interactive editing, retrieval, storage and message distribution of schedules. Since a sender functionality is built into this class, one can actually send such a schedule to ICS, which—following the incorporated start time—lets the DL LCU trigger the ROE and start the communication with NRTS, see `miidevPiezSchedu(3)`. In summary, this provides an engineering interface of executing schedules independent of any OS server, with properly synchronized piezo motion and detector integration if the GEIRS trigger mode has been set to "external."

There is a third way of generation, used by OS to create a schedule on the fly from a few keywords of the `INS.PIEZ` category, `DET.DIT` and `DET.NDIT`. It consists of a filter of these specific keywords in form of `mioSERVER::SetupPreProc` hooks/functions during the SETUP phase, temporarily stored in form of instantiations of the `mioSchedKw` class, and transformed from this representation into schedules with the `operator schedule()` of this class just before the START.

A hybrid form is also provided: one may refer to schedules created interactively (off-line) and stored at well-defined places in the file system with the `INS.PIEZi.FILEi` SETUP keyword. This provides means to let the OS server fetch complicated schedules, which are not construable by the few keywords foreseen in the `mioSchedKw` interface class, without halt requesting for operator intervention.

Bulky frame-by-frame representations of schedules are ASCII dumps (created with the button of `mioSchedu(1)`, Fig. 14) or FITS tables (Sect. 1.5.5).

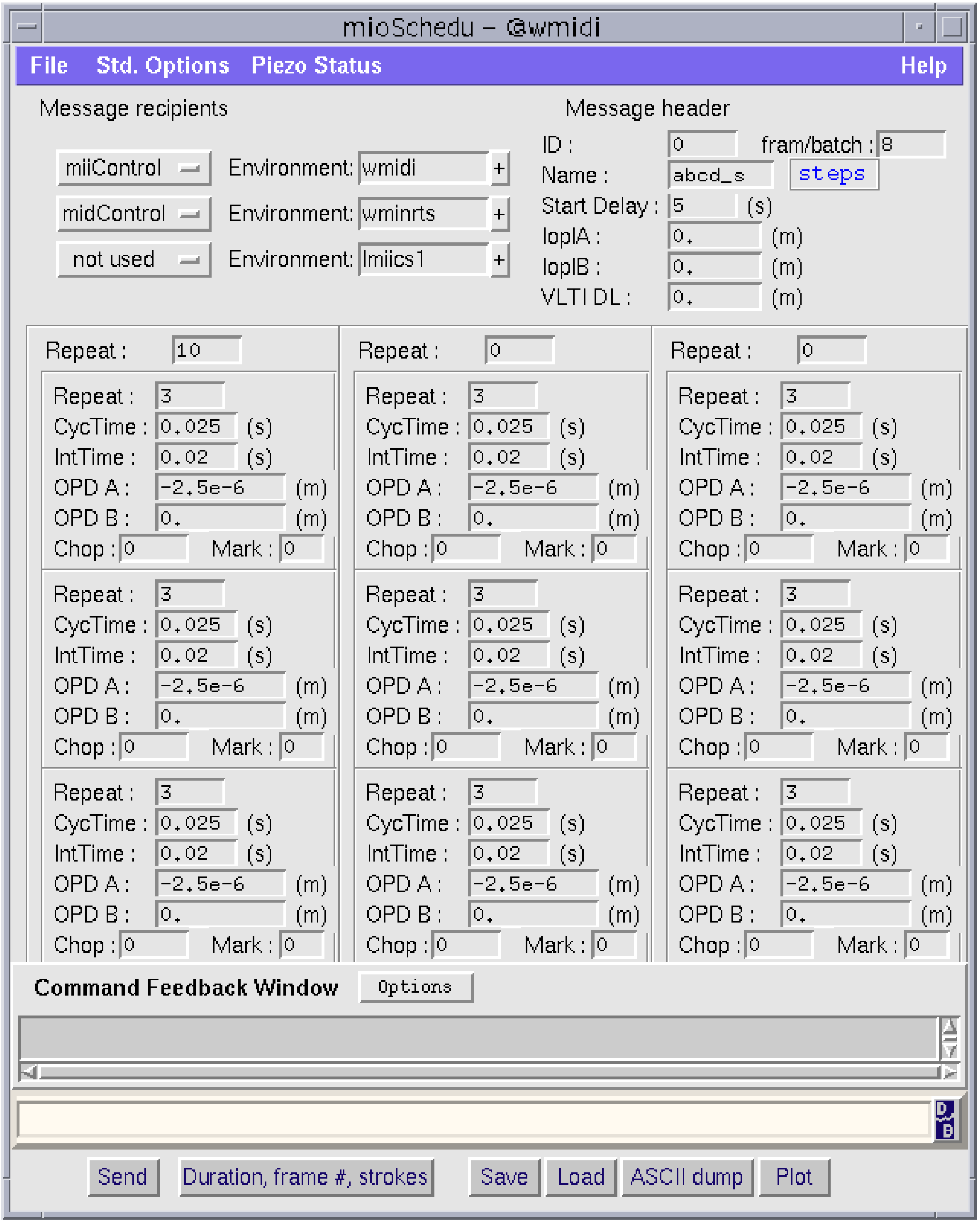



Figure 14: mioSchedu - the Engineering Panel to create and distribute Piezo schedule information to the subsystems. Could be called by pressing the `mioSchedu` button of Fig. 3.

### 3.5 Chopping synchronization

The observation software supports the synchronization between telescope chopping cycles and the detector readout cycles as follows:

- It informs the LCU `lmiics1` at the start of each exposure (that means right after it received itself the `START` command) via a `CHOP` command on the timing parameters (start time, frequency, status,. . . ), which allow the LCU to compute at each time in the future the current chopping phase by looking at its accurate, TIM-based clock. For the detailed set of this information forwarded by the `CHOP` command see the description in `mio/config/mioSpecific.icdt`.
- If chopping is active, it aligns the start time of the first ROE frame (which becomes part of the `SCHEDU` message) to a time in the future which is an integer number of chopping cycles away from the chopping start time. Caution: this is currently done without regard of the `TEL.CHOP.GUIDE`, so depending on the setting of `TEL.CHOP.GUIDE` the first group of frames may be on the sky or on the target.
- The information used to compile the `CHOP` and `SCHEDU` messages is very up-to-date, because OS retrieves it from the scanned ISS OLDB right after it received the `START`, which is (depending on some cpp variables) about seven seconds before the first frame will actually be created by the ROE. It therefore does not matter whether the template scripts continued, interrupted, stopped or re-started the chopping within one of their OB's, because the ISS data base reflects this new information. However, if the template scripts change any of these parameters by talking to ISS *during* an ongoing exposure, MIDI will not notice this and assume that chopping went on periodically as predicted at the `START` time.
- For interactive calculation of chopping SETUP parameters it offers `miopanETC`, Fig. 15.
- for debugging purposes it adds a `COMMENT` with the chopping start time to the FITS primary header—there is no equivalent ISS keyword yet.
- it displays the chopping frequency and status in `miopanControl`, Fig. 5.

This also means

- the observation software has no means to prevent the ROE and the chopping mirrors from being out of phase sooner or later after the first frame integration if the *actual* detector cycle time and the chopping cycle time do not match well.
- and the observation software does not correct or modify are adjust the chopping parameters (presumably sent from the template scripts to ISS) and/or the detector integration time (presumably sent from the template scripts via OS to DCS). However, it supports early assessment of the predicted detector cycle time with `mioCycTime(1)`.
- improving on the precision of the knowledge of the ROE cycle time (as a function of `DET.INT`, `DET.COADD`, `DET.FITSTPL` etc.) is irrelevant to OS, because this would only end up in a more accurate schedule, and this is irrelevant to the LCU `lmiics1` because this steps forward through the schedule as triggered by the ROE, *not* according to cycle times in the schedule.

The implementation is in the `StartPreProcSpecial()` function of the `mioSERVER` class. The description here refers to the implementation *as is*; it does not indicate that one could not think of scenarious where the observation software itself adjusts the chopping frequency (by actively sending commands to ISS) or supports the template scripts by leaving statistical information about the most recent exposure of the past in some OLDB attributes.

## 3.6 FITS files

### 3.6.1 Template files

The man page `mioSetup.fits(5)` describes how the up-to-date list of available file names is obtained. This set ought be the same

- as listed under the `DETFITSTPL` keyword in the ISF,
- the file `MIDI.isf` in `miotsf/config` also shown in Fig. 7,
- displayed with the method described for Fig. 4,
- and the set in the `DET.FITSTPL` selector of Fig. 15.

The option `-midcs` of `mioFitsCnvrt(1)` may be used to generate others in case of need, one may derive new files from existing ones with `fv(1)`, and there may exist IDL routines—but not as part of the OS and therefore not described here—with equivalent functionality.

`mioFits2Win(1)`, with the option `-c` ought be used to check the internal consistency of various pieces in these FITS files. The tests executed on the files are described in the description of the function `check` in the man page `mioCcfits(4)`. Some of them are necessary because the standard [5] demands redundancy of information—often including header keywords that are implicit defined through the tables layout—which may cause discrepancies, others of them are necessary to avoid that GEIRS produces data streams [38] which are not expected by NRTS (which, again, is out of the scope of this manual).

Anyway, the new files should added to the `mio/mio/config/` directory of the `mio` module and distributed by shutting down OS and executing `make all install` in `mio/mio/src`.

Here is an explicit remark concerning rumours that one would have to copy the `mioSetup*.fits` files to wminrts explicitly: The OS server `mioControl` tries to copy these to `wminrts` with `mioSetupDistr(1)` as `mioControl` is started; as a measure against problems that may occur at that time—insufficient file access permissions e.g.—, DCS may do this in addition (see [28]). This installation does not include automatic recognition by BOB or P2PP [9, 3]—the list in the Instrument Summary File must be adapted manually— nor does it prevent NRTS from actually using other files (see [11] for details).

### 3.6.2 Inspecting Data files

Inspection of the FITS data files is supported by `mioFitsListhead(1)`, `mioFitsListstruc(1)`, `mioFitsTablist(1)`, `mioFits2Win(1)`, and `rtd(1)`. `mioFitsCnvrt(1)` offers conversion routines, and these are the basis of the display options of `mioRtd(1)`. The additional software in `mimtoo` is superfluous.

Some of this functionality is bound to buttons in `miopan(1)` (Fig. 3).

`mioCfg2Fits(1)` is called at run-time by the OS server to convert the ICS motor configuration into the binary tables `INS_DESCRIPTION` and `INS_TRAIN`.

### 3.6.3 Postprocessing

After receipt of the FITS data files from NRTS, the binary tables are postprocessed, some or all of the tables `INS_DESCRIPTION`, `INS_TRAIN`, `ARRAY_DESCRIPTION`, `ARRAY_GEOMETRY`, `OPTICAL_TRAIN`,

SCAN_SETUPi are added, and the primary header is constructed "from scratch" based on queries to ICS, DCS and ISS in conformance with the MIDI dictionaries. Some information may be missing depending on the configuration of the subsystems with the various OCS keywords [33]. Postprocessing includes the following modifications of the IMAGING_DATA and IMAGING_DETECTOR tables:

- The GAIN column in IMAGING_DETECTOR is calculated.
- Header keywords ESO DET DID and ESO DET ID are inserted in the IMAGING_DATA and IMAGING_DETECTOR tables.
- The keywords DATE-OBS and MJD-OBS are transferred from the IMAGING_DATA to the IMAGING_DETECTOR table.
- The DETECTOR column in IMAGING_DETECTOR is set to 1.

Some of these action on the FITS files can be switched on or off by modification of preprocessor symbols in C header files. The main reason of disabling this editing the FITS files received by NRTS is that this saves one time-consuming complete reading of the large FITS files. (Experience on Paranal showed that the operators are generally more interested in getting quickly back to the next exposure than generating FITS files that please the Garching DMD community. In this sense, all this polishing of the DCS files is disabled at the moment.)

## 3.7 Troubleshooting and FAQ

### 3.7.1 boss logs non-existing OLDB attributes

**Symptom** The OS server creates log entries of the form

```
Jun 17 12:07:40 te35 syslog: 2003-06-17  12:07:40.999255
   wte35 boss mioControl 214 0 For 'DCS' the dbROOT is set to default:'<alias>MIDI:DCS:DCS'
```

in $VLT_LOG_FILES/logFile, which refer to attributes that do not exist in the IWS CCS database.

**Cause** The OS server base class in the boss module initializes a local variable in bossINTERFACE::SetDefaultDbRoot() at this time.

**Workaround** This log is irrelevant because the configuration keywords of the OCS.DET1 category will supersede this initialization in a later stage of starting mioControl. Either ignore the log or silent this output by reducing the log level (editing OCS.CON.LOGLEVEL in mimcfgINS.cfg).

### 3.7.2 The archived FITS files don't contain the header keywords inserted by DCS

**Symptom** The DCS server(s) delivered binary FITS tables with primary header keywords which seem to be lost on their way into the ESO archive.

**Cause** The OS server inherits the standard boss behavior of creating primary headers from scratch. During the merger processing by the bossArchiver(1), the primary header is lost.

**Solution** The proper way of submitting primary header information is to provide a list of keywords in response to the STATUS command, and to maintain the mimcfgDCS.cfg file, compatible with the dictionaries.

#### 3.7.3 Slow FITS Archival

**Symptom** The time between the final update of the displays on the DCS workstation and OS returning to IDLE seems too long.

**Causes** A copy of a typical 100 MB file to $INS_ROOT/$INS_USER/DETDATA currently needs about 33 seconds (N. Housen, priv. commun July 2004). It is obvious that this occurs at least twice, once for the mioDxf transfer from NRTS, once for re-writing the merged FITS file. In addition, the internet data transfer over the 100BaseT connection for mioDxf needs at least 10 seconds.

**Workaround** The WAIT command should be issued from the sequencer scripts as outlined in the release notes of the VLTCS APR2004, distributed at the standard location in the ESO web pages, which can push some of the FITS processing into the background. Alternatives are to install the $DETDATA directory on one of the two standard, faster 9 GB disks of the IWS, which would require some automated, additional monitoring for the problem decribed in Sec. 2.7.10.

#### 3.7.4 Identical Copies of TIME values in IMAGING_DATA tables

**Symptom** Subsequent values in the TIME column of the IMAGING_DATA table seem to be equal in some cases.

**Causes** If the DL LCU runs without support by the TIM board, it switches to time measurement based on the VxWorks clock, which has a resolution of 10 ms (100 tics per second). These data are transferred via EXPREP messages to NRTS which inserts these into the FITS files.

**Workaround** This is not to be discussed here, because this is rather an ICS issue, and would only occur with some test setups.

#### 3.7.5 UTC and MJD-OBS differ

**Symptom** The primary header keyword UTC of the FITS files is approximately 8 seconds ahead of the time specified by MJD-OBS and OBS-DATE.

**Causes** The header keyword UTC is created by ISS upon receiving a message approximately 7 seconds before the start of the actual exposure. (The value corresponds to the definition of MIO_SERVER_OBS_DATE_DELA in mioSCHED_KW.h.)

**Workaround** This could only be improved by either re-writing part of the OS or within the pipeline code. The simplest solution is to work always with the MJD-OBS and/or OBS-DATE if accurate estimation of the start of the exposure is needed.

# 4 BEYOND THE OBSERVATION SOFTWARE

This section deals with aspects of the MIDI operation which are not due to OS but apparently not covered anywhere else.

## 4.1 Observation Support Software

`miopanETC(1)` is the GUI shown in Fig. 15 which (i) converts three common time formats in the uppermost part, (ii) computes detector cycle times based on DET keywords and the number of detector lines specified implicitly through the FITS template files, and (iii) converts time formats associated with some chopping keywords, optionally bound to the detector cycle time.

## 4.2 FAQ and Troubleshooting

### 4.2.1 Syntax errors with negative motor positions in BOB

**Symptom** Typing numbers like `-.00013` for motor positions in BOB leads to errors of the message system, though the number is perfectly in the valid range.

**Cause** Negative numbers with no digit between the sign and the decimal dot (i.e., in the FORTRAN style) are misinterpreted as option flags by the CCS message parser. A proposal to improve the parser has been rejected by ESO in VLTSW20020405.

**Workaround** Type in negative floating point parameters with an explicit decimal; that would be `-0.00013` in the example above.

### 4.2.2 Negative Integration Delay Time

**Symptom** The keyword `DET DITDELAY` in the primary FITS header, which ought be a positive time measured in seconds, is reported a negative value.

**Cause** If NRTS is run in simulation mode, it may produce `IMAGING_DATA` tables in the FITS files, which are unrealistic because the time difference between the first and last entry in the `TIME` column is smaller than the sum of the `EXPTIME` data. The `TIME` data may even look like a sawtooth function of the `FRAME` data index. Since OS computes `DET DITDELAY` by getting the time interval from the `TIME` column, subtracting the total time of the `EXPTIME`, and dividing by the number of exposures, the value becomes negative.

**Workaround** Ignore and/or request improvement of the simulated NRTS data files.

### 4.2.3 Varying MJD-OBS in a single FITS file

**Symptom** The values of the keyword `MJD-OBS` in the header unit of the `IMAGING_DATA` table and in the header unit of the `IMAGING_DETECTOR` table within a single FITS file differ by about 0.042.

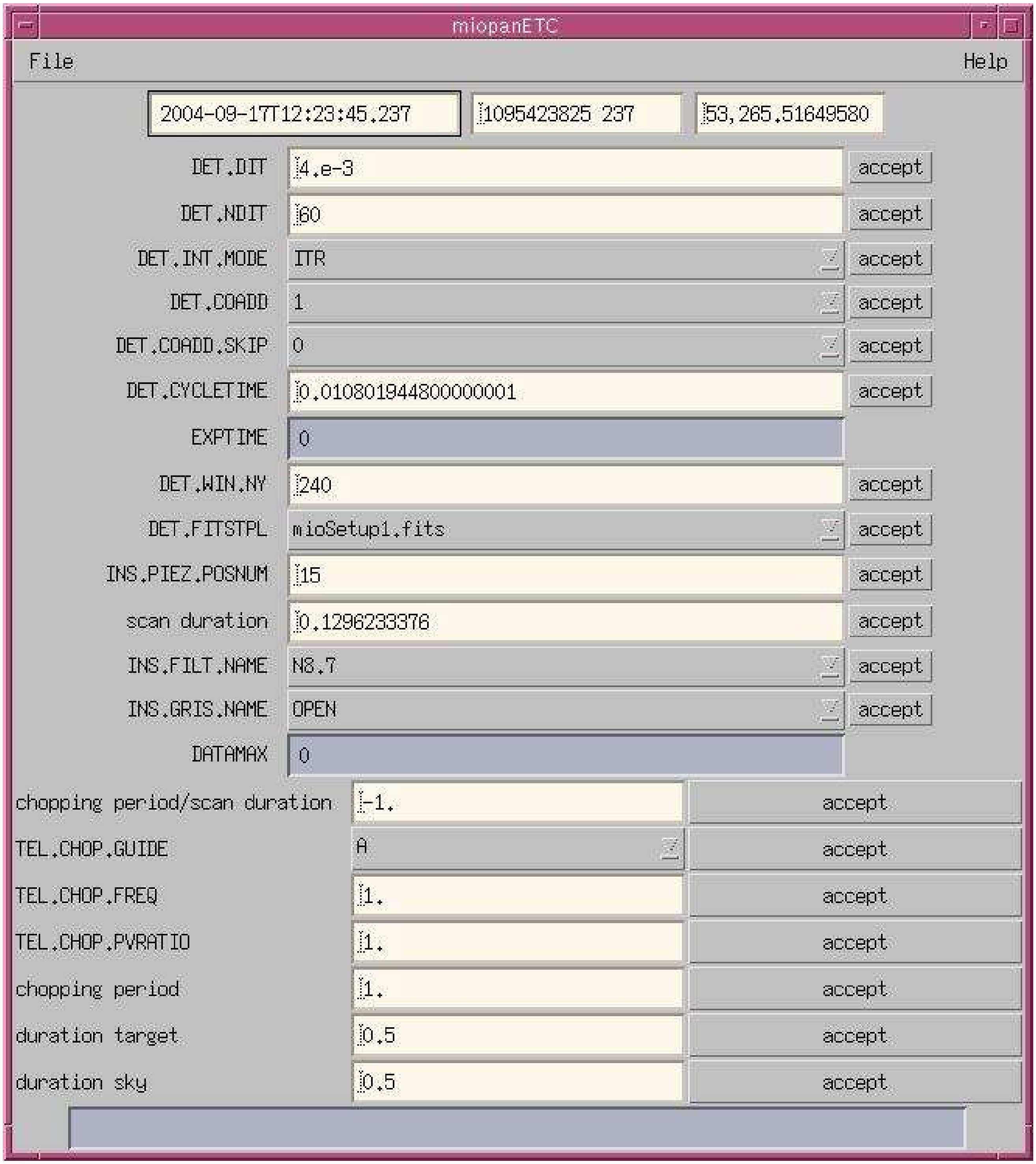



Figure 15: `miopanETC` is a pocket calculator to faciliate the conversion of timing formats.

**Cause** This may happen if NRTS runs on a computer that does not run UNIX in the UTC time zone. The mutual conversion between the `MJD-OBS` and `DATE-OBS` of the `IMAGING_DATA` entries is not yet dealing correctly with the problem considered in VLTSW20020534. Each hour of misinterpretation adds $1/(24 * 3600) \approx 0.042$ to the `MJD-OBS`.

**Workaround** Request improvement of the time handling within NRTS.

### 4.2.4 Cannot start vltisim from the Simulation Control Panel

**Symptom** If one presses the START button of the VLTI Simulation Control GUI shown in [16], which has been opened with the GUI... button of Fig. 2, nothing happens.

**Cause** The START button of the VLTI Simulation Control tries to start the processes in the simulation environment $VLTI_ENVNAME, but does not check that this environment is running. Looking at the $VLTDATA/tmp/logFile, for example with the logMonitor, reveals the error and the failed attempts to connect to the environment.

**Workaround** Write a SPR to request improvement of the vltisim module. Then start the vltisim environment with the STARTUP button of Fig. 2, which calls miinsStart(1) to start the environment, the scan links, and the auxiliary processes, and should result in an ONLINE status in the VLTI column of Fig. 2.

### 4.2.5 FITS compatibility

**Symptom** The format of the binary tables which have names that start with OI_ is largely incompatible with the OIFITS [29, 30] standard.

**Cause** DMD decided not to take action on this issue, see DFS02968. Note that this is out of scope of the OS because these tables are produced by pipeline software.

**Workaround** Ignore the content of these binary tables.

**Symptom** The CHECKSUM values of the FITS files are wrong.

**Cause** As discovered in 09/2005, some part of the pipeline software seems to use an algorithm that computes a CHECKSUM over the entire file, whereas the standard defines values for each HDU. Note that this is out of scope of the OS because OS does not create any CHECKSUM keywords.

**Workaround** Ignore the content of these keywords.

# 5 APPENDIX

## 5.1 Acronyms

**BOB** Broker of Observation Blocks

**CCSlite** RTAP-free version of CCS

**CMM** Configuration Management Module
http://www.eso.org/projects/vlt/sw-dev/wwwdoc/dockit.html

**DCS** Detector Control System

**DL** Delay Line

**DLMT** Delay Line Mirror Translation Stage (of the MIDI warm optics)

**DMD** Data Management Division (of ESO)

**DRO** Detector Read Out

**DWS** Detector Workstation

**ESO** European Southern Observatory `http://www.eso.org`

**FITS** Flexible Image Transport System `http://fits.gsfc.nasa.gov`

**GEIRS** Generic Infrared Software

**GUI** Graphical User Interface

**HDU** header-data unit (of FITS)

**ICS** Instrument Control Software

**IDL** Interactive Data Language `http://www.uni-giessen.de/hrz/software/idl/`

**ISF** Instrument Summary File

**ISS** Interferometric Supervisor-Software

**IWS** Instrument Workstation

**LAN** Local Area Network

**LCU** Local Control Unit

**MIDI** Mid-Infrared Interferometric Instrument `http://www.mpia.de/MIDI`

**NRTS** Near Real-Time Software (of MIDI)

**OIFITS** Optical Interferometry FITS Exchange Format
`http://www.mrao.cam.ac.uk/~jsy1001/exchange/`

**OLAC** On-Line Archive Client

**OLDB** Online data base

**OS** Observation Software

**P2PP** Phase 2 Proposal Preparation `http://www.eso.org/observing/p2pp`

**ROE** Readout Electronics

**TCS** Telescope Control System

**VLT** Very Large Telescope `http://www.eso.org/`

**VLTCS** VLT Control Software

**VLTI** Very Large Telescope Interferometer `http://www.eso.org/vlti`

**VOLAC** VCS OLAC

### 5.2 References

## References

[1] E. Allaert. *VLT Software - HOS/Sequencer User Manual*, 28 March 2002. VLT-MAN-ESO-17220-0737.

[2] E. Allaert. *VLT Software - HOS/Broker for Observation Blocks User Manual*, 24 April 2007. VLT-MAN-ESO-17220-1332.

[3] E. Allaert and A. M. Chavan. *ICD between the VLT Control Software and the Observation Handling Subsystem*, 30 May 2003. VLT-ICD-ESO-17240-19200.

[4] E. J. Bakker. *MIDI observing procedures: user and maintenance manual*, 26 July 2002. VLT-TRE-MID-15824-0263.

[5] P. Ballester and C. Sabet. *VLT Interferometer VLTI Data Interface Control Document*, 3 June 2002. VLT-SPE-ESO-15000-2764.

[6] M. Bec. *VLT Software - Broker for Observation Blocks Software Maintenance Manual*, 28 August 1997. VLT-MAN-ESO-17220-1349.

[7] A. Brighton. *VLT Software Real Time Display User Manual*, 16 May 1999. VLT-MAN-ESO-17240-0866.

[8] F. Carbognani. *VLT Software Environments - Common Configuration User Manual*, 30 October 1998. VLT-MAN-ESO-17200-0855.

[9] F. Comerón and D. Silva. *Phase 2 Proposal Preparation Tool (P2PP), User Manual*, 6 edition, 20 December 2004. VLT-MAN-ESO-19200-1644.

[10] W.D. Cotton, D. Tody, and W.D. Pence. *Binary Table Extensions to FITS*, 1995.

[11] J. de Jong. *MIDI Near-Real Time Software, User and Maintenance Manual*, 8 February 2004. VLT-TRE-MID-15824-0270.

[12] DICB. *Data Interface Control Document*, 1 February 2005. GEN-SPE-ESO-19400-0794.

[13] Alberto Gennai. *Interface Control Document between VLTI Fast Link and Delay Line, OPD Controller, FSU, VINCI*, 15 March 2002. VLT-ICD-ESO-15400-1954.

[14] B. Gustafsson and B. Gilli. *Guidelines for the Development of VLT Application Software*, 29 June 2005. VLT-MAN-ESO-17210-0667.

[15] S. Hippler. *MIDI Computer System Architecture*, 30 January 2001. VLT-TRE-MID-15823-0127.

[16] N. Housen. *Simulation of VLTICS for Instrumentation SW, User Manual*, 1 December 2001. VLT-MAN-ESO-15410-2372.

[17] W. Jaffe. *Overview of MIDI Software Systems*, 31 January 2001. VLT-TRE-MID-15824-0130.

[18] A. Longinotti. *VLT Software, Template Instrument User and Maintenance Manual*, 31 March 2003. VLT-MAN-ESO-17240-1973.

[19] A Longinotti. *VLT Software, INS Common Software, Specification*, 30 January 2004. VLT-SPE-ESO-17240-0385.

[20] A. Longinotti. *VLT Software, VLT Instrumentation Software Specification*, 30 September 2005. VLT-SPE-ESO-17212-0001.

[21] Richard J. Mathar. *MIDI Instrument Control Software*, 1 February 2001. VLT-TRE-MID-15823-0128.

[22] Richard J. Mathar. The unknown target type flag percentage as a function of chopping frequency. Technical report, 3 June 2003.

[23] Richard J. Mathar. MIDI optical path differences and phases. Technical report, 10 March 2006.

[24] Richard J. Mathar. *PRIMA Non-Astrometric Modes, External Fringe Tracking for MIDI*, 29 May 2006. UL-ICD-DDL-15728-0009.

[25] Richard J. Mathar and Walter J. Jaffe. *Software User's Guide and Overview*, 21 September 2004. VLT-TRE-MID-15824-0262.

[26] NASA/Scienc Office of Standards and Technology. *Definition of the Flexible Image Transport System (FITS)*, 29 March 1999. NOST 100-2.0.

[27] Udo Neumann. *MIDI Instrument Control Software, User and Maintenance Manual*, 16 June 2004. VLT-TRE-MID-15823-0266.

[28] Udo Neumann and Clemens Storz. *MIDI Detector Control Software, User and Maintenance Manual*, 20 October 2005. VLT-TRE-MID-15823-0268.

[29] Thomas A. Pauls, John S. Young, William D. Cotton, and John D. Monnier. A data exchange standard for optical (visible/IR) interferometry. In Wesley A. Traub, editor, *New Frontiers in Stellar Interferometry*, volume 5491 of *Proc. SPIE*, pages 1231–1239. Int. Soc. Optical Engineering, 2004.

[30] Thomas A. Pauls, John S. Young, William D. Cotton, and John D. Monnier. A data exchange standard for optical (visible/IR) interferometry. *Publ. Astron. Soc. Pac.*, 117(837):1255–1262, 2005.

[31] D. Popovic. *VLT Software, INS Common Software, Startup Tool User Manual*, 26 August 2005. VLT-MAN-ESO-17240-2153.

[32] D. Popovic. *VLT Software INS Common Software Data Transfer Library User Manual*, 9 February 2007. VLT-MAN-ESO-17240-0637.

[33] E Pozna. *VLT Software, Base Observation Software Stub (BOSS) User Manual*, 7 February 2007. VLT-MAN-ESO-17240-2265.

[34] F. Puech and Ph. Gitton. *Interface Control Document between VLTI and its Instruments*, 11 August 2005. VLT-ICD-ESO-15000-1826. On page 110, $f_c^{-2}$ should read $f_c^{-8/3}$, $f < f$ should read $f < f_c$, and $1.38\,10^{-3}$ should read $1.38 \times 10^{-3}$.

[35] SEG. *Data Flow System High Level User's Guide*, 20 March 2002. VLT-SPE-ESO-19000-1780.

[36] S. Skole and R. Schmutzer. *VLT Software, Installation Tool for VLT SW Packages, User and Maintenance Manual*, 31 December 2005. VLT-MAN-ESO-17240-1913.

[37] S. Stanghellini and A. Michel. Performance of the first two beryllium secondary mirrors of the VLT. *The Messenger*, 94:10–11, 1998.

[38] C. Storz and J. A. de Jong. *ICD Data Transfer from DRO to NRTS*, 15 May 2001. VLT-ICD-MID-15824-0143.

[39] CCS Team. *VLT Software - Central Control Software User Manual*, 31 March 2004. VLT-MAN-ESO-17210-0619.

[40] M. Verola and N. Wolff. *Interface Control Document between VLTI Control System and Delay Line*, 11 December 1997. VLT-ICD-ESO-15400-1542.

[41] A. Wallander. *Final Lay-out of VLTI Control LANs*, 28 May 2004. VLT-SPE-ESO-15410-1957.

[42] A. Wallander. *VLTI Supervisor Software (ISS) Design Description*, 20 July 2005. VLT-SPE-ESO-15410-2114.

[43] A. Wallander. *ICD between VLTI Supervisor SW and VLTI Instrumentation SW*, 17 November 2006. VLT-ICD-ESO-15410-2117.

[44] R. Wilhelm, B. Koehler, and Ph. Gitton. *Functional Description of the VLTI*, 2 August 2002. VLT-ICD-ESO-15000-1918. The radii of curvature of M2, M5 and M7 in Table 4-1 are wrong. See [37] for M2.

**___oOo___**